# FINAL STATUS REPORT

National Aeronautics and Space Administration

Grant NGL 22-009-337

covering the period

August 16, 1972 - February 28, 1973

Submitted by: Paul Penfield, Jr. (Principal Investigator)



May 30, 1973

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Research Laboratory of Electronics
Cambridge, Massachusetts 02139

## SUMMARY OF RESEARCH

### Avalanche Diodes for the Generation of Coherent Radiation

Work under this grant during the last period was concentrated in two areas: solid-state devices and characterization, and optimum imbedding networks for realizing best performance. The BARITT device (Barrier Injection Transit Time Diode) has been under investigation, as reported below. In addition, other work previously reported has resulted in publications, preliminary versions of which are attached as Electrodynamics Memos Nos. 28, 29, and 30.

BARITT diodes are under investigation for possible application as microwave amplifiers and oscillators. Measurements have been made of diode noise figures in the frequency range of 4-6 GHz. Initial results indicate that a noise figure of 6-8 dB may be possible. Devices are under development for operation as amplifiers and sources in the frequency range 2-4 GHz.

We are investigating optimum device structure and fabrication techniques necessary for low noise performance. The possibility of cryogenic operation is being investigated, with a view toward improving the device noise figure.

Models for the device for nonlinear and small-signal operation are under investigation and optimal imbedding and deimbedding models are under consideration.

A list of publications supported by NASA Grant NGL-22-009-337 is attached.

Publications

NASA Grant NGL-22-009-337


E. L. Caples, "Experimental Broadband Mixer," Quarterly Progress Report No. 100, Research Laboratory of Electronics, M. I. T., January 15, 1971, p. 48.

A. Y. C. Chen, "Varactor Frequency Doubler Imbedding Network Analysis and Design," Quarterly Progress Report No. 98, Research Laboratory of Electronics, M. I. T., Cambridge, Mass., July 15, 1970, p. 21.

A. Chu, "Instrumentation – Status of Research," Quarterly Progress Report No. 100, Research Laboratory of Electronics, M.I.T., January 15, 1971, pp. 47-48.

M. Greenspan, K. I. Thomassen, and P. Penfield, Jr., "General-Purpose Microwave Circuit Analysis Incorporating Waveguide Discontinuity Models" (Meeting paper) IEEE Boston Chapter G-MTT Symposium, Boston, Mass., February 28, 1972; IEEE International Microwave Symposium, Chicago, Illinois, May 22-24, 1972.

V. C. Howey, "Electronic Instrumentation," Quarterly Progress Report No. 98, Research Laboratory of Electronics, M.I.T., Cambridge, Mass., July 15, 1970, pp. 21-22.

P. Penfield, Jr., "Proposed High-Efficiency Diode Oscillator," Electronics Letters, Vol. 5, No. 17, p. 387, 1969.

P. Penfield, Jr., "Effects and Modes in Avalanche Diodes," The Avalanche Diode Workshop, New York, December 10-11, 1969.

P. Penfield, Jr., "Computer-Aided Linear-Circuit Design," NEREM 1971 Record, pp. 185-187.

P. Penfield, Jr., "Description of Electrical Networks Using Wiring Operators," Proc. IEEE, Vol. 60, No. 1, pp. 49-53, January 1972.

D. F. Peterson, "Theoretical Small-Signal Analysis of an Avalanche Diode in the s-Plane," Quarterly Progress Report No. 99, Research Laboratory of Electronics, M. I. T., Cambridge, Mass., October 15, 1970, pp. 35-46.

D. F. Peterson, "Intermodulation Distortion in Avalanche Diode Amplifiers," Quarterly Progress Report No. 101, Research Laboratory of Electronics, M. I. T., April 15, 1971, pp. 19-24.

D. F. Peterson and D. H. Steinbrecher, "Circuit Model for Characterizing the Nearly Linear Behavior of Avalanche Diodes in Amplifier Circuits," IEEE Trans. Vol. MTT-21, No. 1, January 1973, pp. 19-27.

D. H. Steinbrecher and D. F. Peterson, "Small-Signal Avalanche Diode Equivalent Circuit Model," Quarterly Progress Report No. 96, Research Laboratory of Electronics, M. I. T., Cambridge, Mass., January 15, 1970, pp. 51-53.

D. H. Steinbrecher, "Optimum Efficiency of a Cascade of Low-Gain Amplifiers," IEEE Trans. Vol. MTT-18, No. 11, November 1970, pp. 951-956.

D. H. Steinbrecher and D. F. Peterson, "Small-Signal Model with Frequency Independent Elements for the Avalanche Region of a Microwave Negative-Resistance Diode," IEEE Trans. Vol. ED-17, No. 10, October 1970, pp. 883-891.

D. H. Steinbrecher, "Low-Noise Microwave Mixers" (Meeting paper) The 1971 IEEE International Convention and Exposition, New York, March 22-25, 1971.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

RESEARCH LABORATORY OF ELECTRONICS

Electrodynamics Memo No. 28 December 21, 1972

# PROPOSED NOTATION AND IMPLEMENTATION FOR DERIVATIVES IN APL*

by

Paul Penfield Jr.**

*This work was supposrted in part by the National Aeronautics and Space administration, NASA Grant NGL 22-009-337.

** Department of Electrical Engineering and Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, Mass. 02139

The lack of notation for derivatives in APL is an important barrier to its acceptance in many disciplines. This paper describes a notation for derivatives and a procedure for accurately evaluating expressions containing derivatives.

## Operators and Functions

In describing APL syntax, it is frequently useful to distinguish functions from operators. Primitive scalar functions, primitive mixed functions, and defined functions are all "functions." The term "operator" is used for a symbol which acts on one or more functions or variables, and produces, as a result, a new function. The new function then acts on its arguments and produces its results, an APL array. The operator, of course, must act before the function can, and therefore, the right-to-left rule for evaluation in APL may have to be violated.

There are three operators implemented in APL: reduction, inner product, and outer product. There has been suggested also a "scan" operator. These operators can, in principle, operate on defined functions as well as primitive functions, although they are not now so implemented. The derivative operator, to be discussed below, can likewise be used on both primitive and defined functions, but the implementation would probably be of little benefit if it were restricted to primitive functions.

## Expectations of this Notation

There are three reasonable expectations for a notation for

derivatives. First, we might expect the notation to denote (and the implementation to calculate) the result of applying the new function produced by the operator, to an argument; that is, to denote the returned array. Second, if the purpose of such an operator is to return another function, one might wish the notation to denote (and the implementation to display) the resultant function. Third, we might expect numerical differentiation of functions defined only by a table of values.

Of these three expectations, the second is difficult, both conceptually and practically, and the third is prone to numerical inaccuracy. The notation and implementation given in this paper fulfill the first expectation but not the others.

One might also expect that it should be possible to devise a notation for integrals. The integral case appears to be much more difficult, however.

## Types of Derivatives

It may not be out of place to recall the wide variety of derivatives that are encountered in mathematically based disciplines. These include the ordinary total derivative, partial derivatives, the substantive derivative of fluid mechanics, and the gradient, divergence, and curl in field theory, as well as high-order derivatives such as scalar and vector Laplacians. Derivatives are taken with respect to any argument of a function, or with respect to any parameter.

The proposed notation not only covers all these cases, but can be further generalized. Differentiation is allowed not only

of functions with respect to arguments, but also of functions or expressions with respect to global variables.

## Graphics

Two new symbols are required for the derivative notation, one for an operator and the other for a function. The symbols ⍁ and ⍂ are used here, without any implication that they are optimum.

## Notation for Derivatives of Monadic Functions

Consider first a monadic primitive scalar function, with a scalar argument. The syntax for this function is `Z←F A`; the derivative of `F` with respect to `A`, evaluated at the particular value of `A`, is denoted `F⍁A`. For example, `+⍁3` is equal to `1`; `*⍁2` is equal to `¯0.25`; and `⌊⍁3.5` is equal to `0`.

Next, consider any monadic function, such as a primitive scalar function, a primitive mixed function, or a defined function. Again, the syntax is `Z←F A`. The derivative of `F` with respect to `A` is denoted `F⍁A`, and is defined provided `A` and `F A` are conformable. The rule for conformability is the same as the rule for the expression `A+F A`; that is, either `A` and `F A` have the same dimensions, or else one of them has only one element. If `A` is a scalar and `F A` is an array, then `F⍁A` consists of the derivative of each element of `F` with respect to `A`, evaluated, of course, at the value of `A` given. For example, if the velocity `V` of a particle and its position `R` are both functions of time `T`, and velocity and position in

3-space are represented by vectors of length 3, then the velocity is the derivative of the position, as expressed by the equation `(V T) = R⍁T`.

If `A` and `F A` have the same dimension, then `F⍁A` consists of the partial derivatives of each element of `F A` with respect to the corresponding element of `A`. In general, each element of the results depends on the values of all the elements in `A`, and is, of course, evaluated for the `A` in question. This form of the derivative is useful, for example, in describing waveforms as a function of time: if `F T` is the function `2×*¯5×T` which might be the natural response of a first-order physical system, then `F⍁T` would calculate an array consisting of the derivative of `F` evaluated at each of the times in the array `T`.

Finally, if `F` is a scalar function of a vector argument `R` which might stand for position in space, then `F⍁R` is a vector, each element of which is the partial derivative of `F` with respect to the corresponding element of `R`; that is `F⍁R` is the gradient of `F`. Thus, the relationship between electric field vector `E` and potential `PHI` in electrostatics can be written in the form `E = -PHI⍁R`. As another example, `(×/⍁A)=(×/A)÷A` except that it works properly even if `0∈A`. In the last example, note that the reduction operator acts before the differentiation operator.

## Outer Derivatives of Monadic Functions

It is frequently necessary to denote the partial derivatives of each element of the result of a function, with respect to

each element of its argument. This concept resembles that of the outer product, and a similar notation is suggested. Thus, for a monadic function $F$, $F\circ.\natural A$ is called the "outer derivative". There are no conformability requirements; the outer derivative has dimension $(\rho F\circ.\natural A)=(\rho F\ A),\rho A$. An example of the outer derivative is the velocity-gradient tensor in fluid mechanics, of which the rotation is the antisymmetric part and the strain rate is the symmetric part. Thus, if the velocity $V$ is a function of space $R$, then the rotation is $0.5\times(V\circ.\natural R)-\lozenge V\circ.\natural R$ and the strain-rate tensor is $0.5\times(V\circ.\natural R)+\lozenge V\circ.\natural R$. As another example, $+\circ.\natural\iota N$ is the unit matrix of size $N,N$.

Inner Derivatives of Monadic Functions

The inner derivative of a function $F$ is denoted $F+.\natural A$, and the conformability requirements are the same as for the inner product $(F\ A)+.\times A$. As an example, one of Maxwell's equations in electromagnetic field theory states that the divergence of the magnetic field $B$ is zero, or $0=B+.\natural R$.

Generalizations of the inner derivative are possible by using scalar dyadic functions other than $+$. For example, $F\times.\natural A$. For a function $F\ T$ of time, its largest derivative at any time in a vector $T$ is given by $F\lceil.\natural T$.

High-Order Derivatives of Monadic Functions

Second and high-order derivatives of monadic functions are denoted similarly, with multiple use of the derivative

operator, for example `F⍁⍁T`. Thus, `F∘.⍁∘.⍁A` is the collection of all possible second-order partial derivatives of `F`, and has dimension equal to `(⍴F A),(⍴A),⍴A`.

An interesting second-order example is the Laplacian of a scalar function `F` of space coordinates `R`, which is `F∘.⍁+.⍁R`. Note that the outer derivative is performed before the inner derivative; that is, the derivative on the left before the derivative on the right, in contrast to the normal APL convention about order of execution of functions. If the function `V R` is a vector function of `R`, then the vector Laplacian is `V∘.⍁+.⍁R`. As another example, consider Newton's law for a particle, where `F` is the force, `M` is the mass, and `A` is the acceleration, the second derivative of its position `R` with respect to the vector of times `T`: `F=M×R⍁⍁T`. As another example, the homogeneous differential equation obeyed by a series RLC electrical network: `0=(L×I⍁⍁T)+(R×I⍁T)+(I T)÷C`.

## Derivatives of Dyadic Functions

The notation for derivatives, inner derivatives, and outer derivatives of dyadic functions is similar to that for monadic functions. Consider a dyadic function `F` with syntax `Z←A F B`. The derivative of `F` with respect to `B`, denoted `A F⍁B`, must satisfy the conformability constraint suggested by the operation `B+A F B`. The result is the derivative (or partial derivatives) with respect to `B` or its elements, keeping `A` constant. The result is evaluated for the particular values of `A` and `B` presented.

A dyadic function can also be differentiated with respect to its left argument. In this case, the symbol ⍤ is moved to the left (that is, it appears between the function name and the independent variable). The rules are similar and in particular, for the outer derivative, its dimension is $(\rho A \circ .⍤F\ B)=(\rho A),\rho A\ F\ B$.

As an example, if functions for current density $J$ and charge density $Q$ are dyadic, with time on the left and space on the right, then the equation for conservation of charge, which appears in conventional notation as $\nabla \cdot J + \frac{\partial q}{\partial t} = 0$ would appear in the new notation as $0=(T\ J+.⍤R)+T⍤Q\ R$. In a similar way, the substantive derivative, or "total derivative" of fluid mechanics $\frac{D}{Dt} = \frac{\partial}{\partial t} + V.\nabla$ can be applied to a dyadic function $N$ of space and time: $(T⍤N\ R)+(T\ N\circ .⍤\ R)+.\times V$.

Some other interesting examples: The derivative of the maximum function $A\lceil ⍤B$ is equal to $B\geq A$. The derivative of the times function with respect to one argument is the other argument, that is $(A⍤\times B)=B$. If $A$ is a vector then $A\times \circ .⍤A$ is a diagonal matrix with the elements of $A$ along the main diagonal.

## Derivatives of Expressions

The notation so far is suitable for differentiation of a function with respect to an explicit argument. It may be desired to differentiate a function with respect to a global variable, or to differentiate an APL expression with respect

to some variable in the expression. This can often be denoted by defining an extra function with the desired independent variable as an argument, but that technique is neither convenient nor natural. A slight generalization of the preceding notation is useful.

The derivative of an APL expression $E$ with respect to a variable $X$ is denoted $E⍁X$. If the expression is more complicated than a single niladic function, it must be parenthesized. The variable on the right must be a variable name, perhaps indexed, but not a constant or an expression. The expression might involve the independent variable explicitly, as in the identity $(N×A*N-1)=(A*N)⍁A$; or perhaps as a global variable in a function, as in the identity $((A×B)⍁X)=((A⍁X)×B)+A×B⍁X$; or perhaps in both roles.

Outer and inner derivatives of expressions with respect to global variables are denoted similarly, $E∘.⍁X$ and $E+.⍁X$.

Note that since $X$ must be a variable name rather than an expression, the following examples are syntax errors: $E⍁X+2$; $E⍁(X)$; $E⍁-X$; $E⍁X,0⍴C←2$; $E⍁X←3$.

## Implementation

The implementation of this notation appears to be straightforward, although perhaps demanding of space and time. The basic idea is that during execution of a function (or expression) being differentiated, a more complicated procedure is used. Every execution of a primitive function is accompanied by the calculation of the corresponding derivative. Thus, as

the defined function is executed, line by line, not only are the values of the expressions calculated, but also their derivatives, always with respect to the original value of the argument in question. When the function finally returns, the value of the return variable is discarded and only the derivative is used.

This procedure, which is defined in detail below, in principle can work for any number of derivatives of arbitrary order.

## Details of the Implementation

A new system-dependent function which I will denote here as I50, is required. Initially, this is set to the empty vector ⍬.

Consider a function $F$ whose argument $A$ is the independent variable, with respect to which the differentiation is being carried out. When the execution of $F$ begins, several things happen. First, the value for I50 ceases to be the empty vector and becomes the vector 0,1. Second, the independent variable is accompanied by 1∘.=A, that is, the outer derivative of $A$ with respect to itself. Other variables, both global and local, are also considered to be accompanied by their partial derivatives with respect to $A$, in every case an array of zeros. Thus, all variables, both global and local, are considered to be not only their values but also their outer derivatives with respect to $A$.

As the function is executed, line by line, each execution

of a primitive function is accompanied by the evaluation of the derivative of the result with respect to *A*; thus, for every array generated there is also generated another array consisting of the outer derivative of that quantity with respect to *A*. The chain rule of differentiation is used. Naturally, the derivatives so calculated will all be zero until *A* actually enters into the calculation.

If another defined function is called within the function being differentiated, this mode of operation continues and when this function terminates execution, not only the value of the return variable but also its outer derivative with respect to *A* will be returned.

A given differentiation is completed when the function (or expression) being differentiated finally returns, or else is removed from the state-indicator stack by the right arrow. When the function returns, the return variable has both a value and its outer derivative with respect to *A*. The value is discarded and the derivative is returned. In the case of the derivative or inner derivative, appropriate operations are performed to extract the desired derivative. At the same time, I50 is reset and all global variables have their accompanying derivatives erased.

When more than one derivative is being considered, a similar process applies. High-order derivatives with respect to the same argument cause values of 2 or more to be inserted into I50, rather than 1, and variables are augmented with both

the first and second derivative with respect to *A*.

Derivatives with respect to other variables may be encountered while any given derivative is pending. In that case, ⎕I50 will contain not only the 1 (or higher number) associated with the initial derivative, but also a number associated with the more recent derivative. Thus, ⎕I50 is a push-down stack, with one entry corresponding to each independent variable. During execution of functions, not only the derivatives with respect to each of the independent variables, but also cross partial derivatives have to be calculated. High-order derivatives of any order can be calculated recursively by using only the formulas for first-order derivatives.

While a derivative is being calculated, that is, while ⎕I50 is not empty, the normal APL environment is changed in a few ways. This can be detected while the function is executing, or while it is suspended. First of all, the state of any pending derivatives can be interrogated by asking for ⎕I50; ⍴⎕I50 gives the total number of different independent variables under consideration, and +/⎕I50 gives the highest-order derivative being calculated. Second, execution of the system commands *)SI* or *)SIV* will cause an extra symbol ⍏ to be printed next to the names of functions being differentiated. Third, the derivative of any expression *E* with respect to the Nth independent variable is denoted by *N⍏B*, or for the outer derivative, *N∘.⍏F*, or for the inner derivative, *N+.⍏E*. This feature allows functions to branch according to the values of derivatives, and also allows the display of derivatives for debugging.

The symbol ⍷ can also be used monadically, as in ⍷*E* or ∘.⍷*E* or +.⍷*E* to denote the derivative with respect to the most recent independent variable.

Needless to say, much space and time will be saved by adopting the strategy of not actually storing derivatives that are identically zero. Thus, if execution is suspended while a derivative is pending, independent calculations can be carried out without necessarily calculating all the derivatives which end up equal to zero anyway.

It may sometimes be desired to obtain both the result of a given function and its derivative without re-executing it, for example in case the function uses terminal input or makes changes in global variables. This is easily accommodated with the proposed notation. The last time the return variable is assigned, also assign some global variable. When the function returns, the global variable will retain the value of the function whereas the return variable will return the derivative.

Unresolved Problem

The derivative of every primitive function in APL can be written in terms of other primitive APL functions by use of the chain rule, with one exception. The exception is the scalar function !. The derivative of the gamma function cannot be expressed in terms of other implemented APL primitives. Furthermore, it is not known whether derivatives of arbitrary orders can be calculated through a recursion formula.

## Conclusions

The notation for derivatives in this paper is convenient, yet general enough to cover probably all instances of differentiation in mathematics. It does not, however, perform or denote numerical differentiation, nor does it indicate the display of functions which are derivatives of other functions. The implementation outlined here is believed to be feasible, and if realized, would lead to a significantly more useful APL.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

RESEARCH LABORATORY OF ELECTRONICS

Electrodynamics Memo No. 29 March 30, 1973

# TIGHTNESS: A NEW ATTRIBUTE OF TWO-PORT NETWORKS*

by

Paul Penfield Jr.**

**Abstract:**

The degree to which a two-port network tightly couples its input to its output is given a quantitative measure. This attribute is different from reciprocity, passivity, stability, symmetry, and other two-port-network attributes.

---

* This work was supported in part by the National Aeronautics and Space Administration, NASA Grant NGL 22-009-337.

** Department of Electrical Engineering and Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge Massachusetts 02139.

## Introduction

Several attributes of two-port networks are well known, including reciprocity, passivity, losslessness, unconditional stability, and symmetry. Not all two-port networks have all these attributes, of course, and two-ports can be classified according to whether each attribute is or is not obeyed. For example, we have the well known class of reciprocal networks, or unconditionally stable networks. Many of these attributes can also be defined for one-port or n-port networks. Each can be expressed as a constraint on the two-port parameters, for example reciprocity implies $Z_{12} = Z_{21}$ and symmetry implies, in addition, $Z_{11} = Z_{22}$. Each of the above attributes can be associated with a numerical factor that is either in a certain range or equals a certain value if the attribute is present. Other information can sometimes be conveyed by the value of the factor even if the attribute is not obeyed. For example, the "reciprocity factor" $Z_{21}/Z_{12}$ is equal to 1 for reciprocal networks, and if it is different from 1, its magnitude suggests the degree to which the two-port is unilateral, with values of $\infty$ or 0 for the unilateral case. Similarly, the "symmetry factor" $Z_{11}/Z_{22}$ for reciprocal two-ports is 1 for symmetric networks, but if it is not 1, its value indicates how the impedance level is transformed by the two-port.

## Tightness

A new attribute, "tightness," is motivated by the observation that the two networks in Figure 1 are somehow basically different, yet their difference is not suggested by any of the attributes listed above. (Note that they are both reciprocal, passive, lossless, unconditionally stable, and symmetric.)

A tight two-port network is, intuitively, defined here as one for which changes in the termination at the input or the output are greatly visible from the other port. A network that is not tight in this sense is one in which the output is more or less decoupled from the input, or vice versa. A useful numerical measure of the tightness of the network is found by comparing the input impedance $Z_{in}$ for two different load impedances. For the two loads it is convenient to choose open and short circuits, and the "tightness factor" TF is defined as the ratio of the open-circuit impedance to the difference of the impedances. That is,

$$TF = \frac{Z_{oc}}{Z_{oc} - Z_{sc}} \qquad (1)$$

A simple evaluation gives this in terms of the Z or Y or ABCD parameters as

$$TF = \frac{Z_{11}Z_{22}}{Z_{12}Z_{21}}$$

$$= \frac{Y_{11}Y_{22}}{Y_{12}Y_{21}}$$

$$= \frac{AD}{AD - BC}$$

Parker, Peskin, and Chirlian[1-6] have observed that the ratio $Z_{oc}/Z_{sc}$ (and therefore the tightness factor) is independent of which port is regarded as the input.

A tightly coupled network has a value of TF close to 1, and a network in which either the input or the output is decoupled from the other has a tightness factor of ∞. This includes networks in which the input and output are unconnected, like Figure 1-b, as well as unilateral networks. A gyrator has a tightness factor of 0.

The tightness attribute, like the others mentioned above, can be tested at any frequency, and the tightness factor can be evaluated as a function of frequency, either numerically or analytically. It is possible for a network to be tight at some frequencies, but not at others.

## Examples

The tightness factors for several simple two-port networks are given in Figure 2. Note that in the case of a mutual

inductor, the tightness factor is $1/k^2$ where k is the coupling coefficient. In the examples the tightness factors are all real, although in general complex values should be expected.

## Discussion

This concept resembles in some ways that of compactness[7-10], but is slightly different, and can be applied to any linear two-port, whereas compactness is usually restricted to lumped, or lossless, or RC networks. To relate these two concepts, a network is compact if it is tight at all its natural frequencies.

Anderson and Ku[5] have discussed the ratio of open-circuit to short-circuit impedances for n-ports, and their approach might be useful as a basis on which to extend the tightness concept to n-ports.

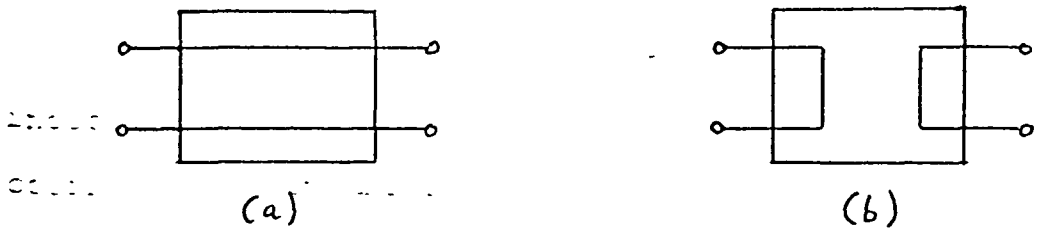



Figure 1. Two extreme examples of two-port networks. Network (a) is tight whereas network (b) is not.

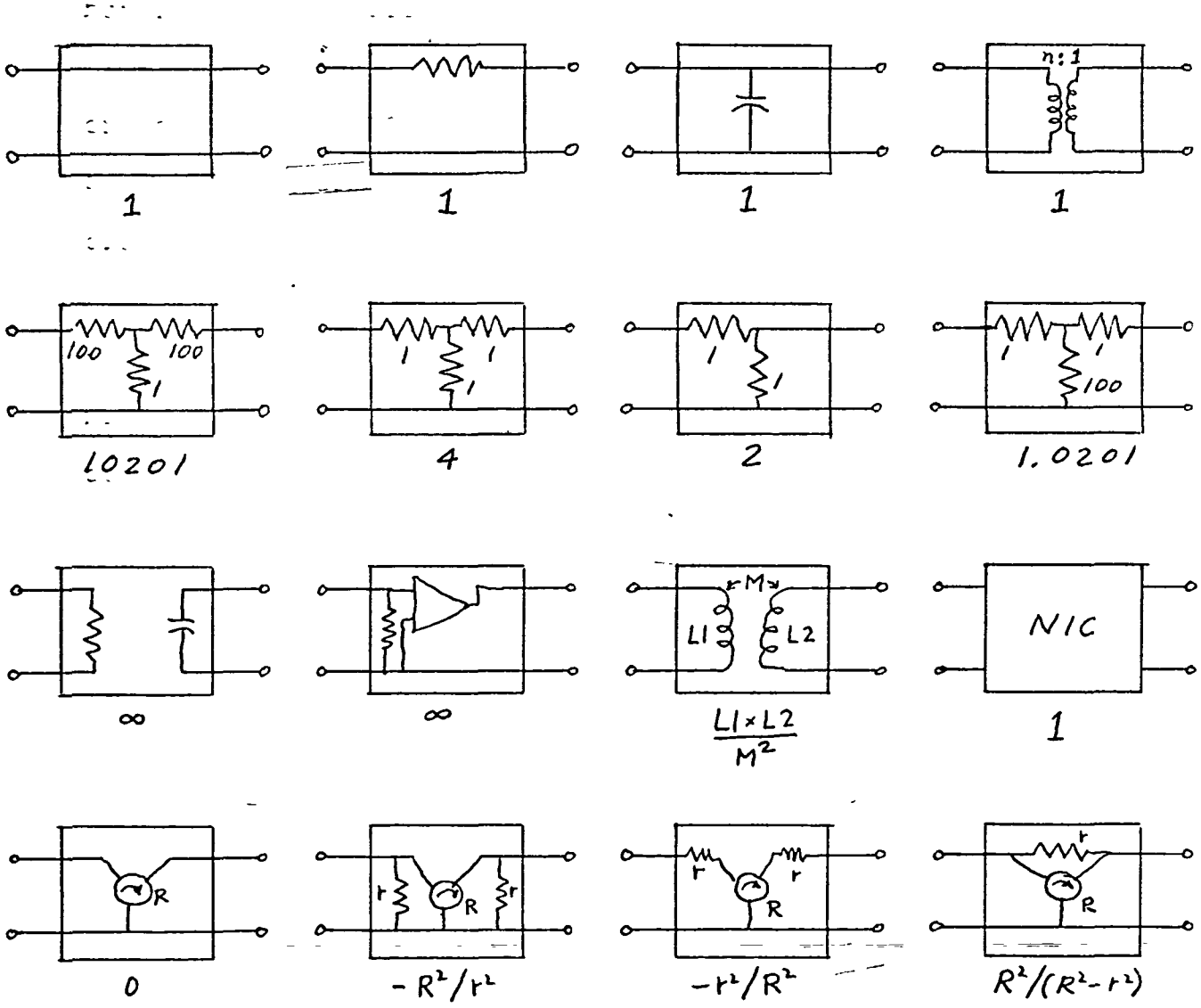



Figure 2. Values of the tightness factor for several two-port networks.

## References

1. S. R. Parker, E. Peskin and P. M. Chirlian, "On the Invariance of the Ratio of Open and Short Circuit Impedances in Linear Networks," Proc. IEEE, vol. 53, no. 7, pp. 760-761; July, 1965.

2. M. H. Crothers and G. H. Fett, "On the Invariance of the Ratio of Open- and Short-Circuit Impedances in Linear Networks," Proc. IEEE, vol. 54, no. 2, pp. 318-319; February, 1966.

3. M. F. Moad, "On the Invariance of the Ratio of Open- and Short-Circuit Impedances in Linear Networks," Proc. IEEE, vol. 54, no. 5, p. 817; May, 1966.

4. W. M. Van Loock, "The Invariance of the Ratio of Open- and Short-Circuit Impedance in Linear Networks, and a Relation for the Reflection Coefficient," Proc. IEEE, vol. 55, no. 2, pp. 232-233; February, 1967.

5. J. Andersen and W. H. Ku, "A General Invariance of the Ratio of Open- and Short-Circuit Impedances for Linear n-Port Networks," Proc. IEEE,vol. 55, no. 7, pp. 1223-1224; July, 1967.

6. P. Penfield, Jr., R. Spence, and S. Duinker, "Tellegen's Theorem and Electrical Networks," The M.I.T. Press, Cambridge, Massachusetts; 1970; Section 5.12.

7. B. J. Dasher, "Synthesis of RC Transfer Functions as Unbalanced Two Terminal Pair Networks," IRE Trans. on Circuit Theory, vol. CT-1, pp. 20-34; December, 1952.

8. I. Cederbaum, "On the Residues of the Impedance or Admittance Matrices of n-ports," IRE Trans. on Circuit Theory, vol. CT-4, no. 1, pp. 20-21; March, 1957.

9. P. Slepian, "On the Compactness of an RC Quadripole," IRE Trans. on Circuit Theory, vol. CT-5, no. 1, pp. 61-65; March, 1958.

10. H. B. Lee, "The Physical Meaning of Compactness," IEEE Trans. on Circuit Theory, vol. CT-10, no. 2, pp. 255-261; June, 1963.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

RESEARCH LABORATORY OF ELECTRONICS

Electrodynamics Memo No. 30 April 25, 1973

# *MARTHA**

by

Paul Penfield Jr.**

---

* This memo is a preliminary version of a chapter in the forthcoming book, "Handbook of Circuit Design Languages", R. W. Jensen and L. P. McNamee, editors, Prentice-Hall, Inc. Early parts of the development of *MARTHA* were supported in part by the National Aeronautics and Space Administration, NASA Grant NGL 22-009-337.

** Department of Electrical Engineering and Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139.

Contents

# I. INTRODUCTION

*MARTHA* is a notation for denoting electrical networks, and it is also a computer program which uses this notation.

## A. General Description

In *MARTHA*, the notation, every network is either an element (resistor, capacitor, transmission line, etc.) or else one or two previously defined networks wired together. The basic idea is illustrated in Figure 1. In this example, *S* and *P* are "wiring functions" and *R*, *L* and *C* are "element functions". *S* and *P* are dyadic; *R*, *L* and *C* are monadic. The functions used in Figure 1 are sufficient to describe any series parallel network containing linear resistors, capacitors, and inductors. For more complicated networks, other, similar, functions in *MARTHA* are used.

As a notation, *MARTHA* is an alternative to the widely used schematic diagram. It can be used for communication from one person to another (for example, for documentation purposes), or from a person to a computer, or from a computer to a person. It is useful both in analysis programs (where the user writes it and the computer reads it) and in synthesis programs (where the computer writes it and either the computer or the user reads it).

*MARTHA*, the computer program, is a network-analysis program imbedded in the interactive language APL, and the syntax

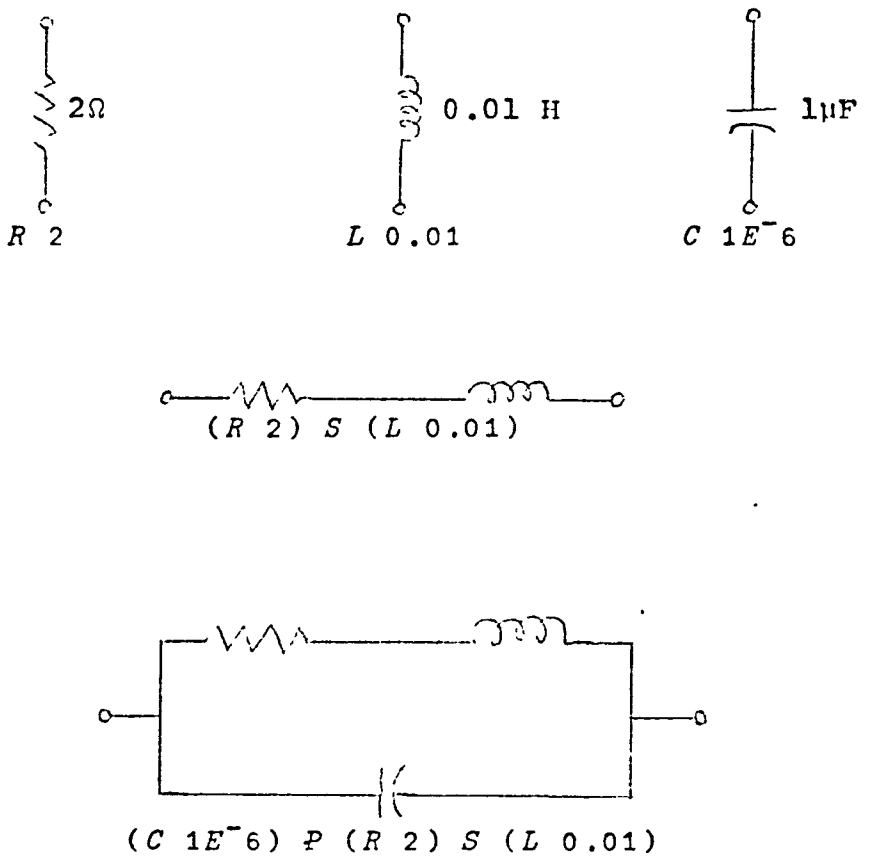



Figure 1. Illustration of *MARTHA* notation for a model of a parallel tuned circuit. The functions *R*, *L*, and *C* define elements, and the functions *S* and *P* perform the wiring. The functions *R*, *L*, and *C* are "monadic" (having one argument, located on their right), and *S* and *P* are "dyadic" (having two arguments, one on each side). These five functions are sufficient to denote any linear series-parallel RLC network.

resembles that of APL in many ways. The simplicity and versatility of the input language make *MARTHA* relatively easy to use. It differs from most of the other programs in this book in that it is naturally oriented toward ports rather than toward nodes. In *MARTHA* node matrices are never calculated or used, so that time-consuming matrix inversion is avoided altogether. There are no difficulties associated with capacitor or inductor loops or tiesets, and it is never necessary to find the eigenvalues of any large matrix. Analysis is carried out at all frequencies simultaneously, using APL's fast array handling.

## B. Capabilities

The program *MARTHA* performs frequency-domain analysis of 1-port or 2-port networks which are made up of 1-port or 2-port linear elements wired together so that at every stage in the construction, only 1-ports and 2-ports are used. Examples of networks of this type are most amplifiers, filters, and microwave systems. Networks may be active or passive, may be reciprocal or nonreciprocal, and may be lumped or distributed (or a combination of both).

Engineers may use *MARTHA* for circuit analysis without knowledge of APL. The simple and uniform notation in *MARTHA* is advantageous to beginners, as is the interactive nature of *MARTHA*. However, users who know APL can write their own func-

tions to control *MARTHA*. For example, network definitions, including both parameter values and topology, can be varied under program control. Advanced users can write synthesis algorithms using *MARTHA* notation for the resulting network. In other words, *MARTHA* can be used as a programming language besides simply as a program.

Although *MARTHA* is intended for general-purpose network analysis, different users can make use of rather specialized portions of *MARTHA* to make up their own special-purpose analysis and synthesis systems.

Microwave engineers can use the distributed elements in *MARTHA*, and wave variables (with complex, frequency-dependent normalization if desired), and Smith-chart plotting. Aspects of interest to amplifier designers include various transistor models, calculation of several measures of gain and stability, and *MARTHA*'s ability to plot Nichols charts and the U/A "gain plane". Filter designers can make use of *MARTHA*'s ability to scale frequency and impedance, and to perform high-pass and band-pass transformations. For active filters, there are four operational-amplifier models of differing complexity. Of interest to experimentalists is *MARTHA*'s ability to work with tables of measured performance, and use them for calculation and interpretation as elements. Engineers concerned with model making will find an extensive repertoire of basic elements including 16 controlled sources, negative-impedance convertors, gyrators, and elements whose response goes with complex frequency to an integral power.

The user can define many different networks, and wire these together or analyze them when desired. Two or more networks can be analyzed at the same time, and the results compared. There are more than a hundred different response functions that can be requested. The results of an analysis can be printed, or plotted vs frequency, or vs any network parameter (on linear or log scale), or vs another response. The response of one network can be plotted against the response of another network, or against a numerical table of values. Alternatively, the results can be stored for later display or calculation, possibly using other results.

*MARTHA* incorporates an extensive set of tools for defining, editing, manipulating, and interpreting tables of numerical values. Such a numerical function of frequency (*FOF*) can be interpreted as a quantity to be printed or plotted (alongside the results of a normal *MARTHA* analysis) or as the impedance, admittance, or scattering coefficient of a numerically defined element.

## C. Documentation

*MARTHA* is described in the book, Paul Penfield Jr., "*MARTHA* User's Manual", The MIT Press, Cambridge, Mass., 1971. That book completely covers versions of *MARTHA* dated 71. For versions of *MARTHA* dated 73 (and this includes all versions on commercial time-sharing computers)

several improvements are described in the pamphlet, Paul Penfield Jr., "*MARTHA* User's Manual, 1973 Addendum, The MIT Press, Cambridge, Mass., 1973. Users of *MARTHA* can determine their version date by referring to the line which starts *CIRCUIT ANALYSIS BY MARTHA* at the top of each print or plot: either 71◦ or 73◦ appears in that line.

Several other publications describing *MARTHA* or the ideas behind it appear in the bibliography at the end of this chapter. See references 1 - 4.

Besides these publications, there is extensive on-line documentation in *MARTHA*; for information on how to access this, type

```
)LOAD 100 HOWMARTHA
DESCRIBE
```

## D. Availability

*MARTHA* is available both from commercial time-sharing computer companies, and for use on separate machines that run APL. Inquiries should be directed to the Manager of Software Services, The MIT Press, 28 Carleton Street, Cambridge, Mass. 02142.

## II. PROGRAM STRUCTURE

In APL, programs are stored in "workspaces". *MARTHA* consists of nine workspaces of which one is purely documentation, and seven constitute the "*MARTHA* library". The basic workspace*, 100 *MARTHA*, contains about 70 cooperating "foreground" APL functions that the user may call directly, several "background" utility functions called by the "foreground" functions, and a few global variables. The foreground functions fall into six categories. First are functions which create elements, such as the functions *R*, *L* and *C*. Second are functions which wire networks together, for example *S* and *P*. Third are functions that calculate the response of the network, for example impedance, admittance, reflection coefficient, or VSWR of one-port networks, or any of the two-port parameters or various gain or stability measures of two-port networks. Fourth are functions which can modify the response functions by taking the real part, imaginary part, magnitude, etc. Fifth are functions that help specify the format of the output. Finally, are some miscellaneous functions to aid in defining networks.

The *MARTHA* library contains additional more specialized, functions in these same categories. It is a standard feature of the APL workspace-storage system that individual functions, or groups of functions, can be copied into the user's active workspace; by so doing the user can select those functions from the library that he needs and leave the rest behind, thereby creating his own, personalized, version of *MARTHA*.

---

* On some computers, the number in the workspace names is different from 100.

## III. NETWORK ELEMENTS

Table I and Figure 2 show the network elements defined in *MARTHA*. The two-port networks *WR* and *WTHRU* are constants, and all the others are monadic functions. The simple functions *R*, *L*, and *C* (resistors, inductors, and capacitors) have been illustrated earlier. The function *L*, when used with an argument of length 3 (that is, a vector with 3 numbers in it), produces a mutual inductor; thus the function *L* produces different elements according to the length of its argument. The same is true of the functions *TEM*, *WG*, and *OPAMP*. The waveguide function *WG* produces a length of waveguide if its argument has 3 numbers in it, but if only 2 are present (the length is absent) the result is the frequency-dependent characteristic impedance of the waveguide. The waveguide analysis is valid both above the cutoff frequency, where the characteristic impedance is real, and below the cutoff frequency, where it is imaginary. Examples:

*R* 40 (resistor, 40 ohms)

*L* .015 (15-mH inductor)

*L* .015 .02 .01 (2-port mutual inductor)

*IT* 3 (ideal transformer, turns ratio 3:1)

*WG* 1*E*9 377 (matched load of waveguide with cutoff frequency 1 GHz, 377-ohm characteristic impedance at f = ∞)

*WG* 1*E*9 377 .6 (60-cm length of waveguide)

*WG* 1*E*9 377 90 *DEGREESAT* 2*E*9 (quarter-wave section of guide)

*WG* 1*E*9 377 .4 *FORDIEL* 2.5 (dielectric-filled guide)

Table 1. Elements defined in *MARTHA*. In the equations, *S* is j2πf. The library workspace 100 *MARTHAE* contains many additional elements.

| ELEMENT | TYPE | NAME | ARGUMENT VECTOR | REQUIRED | EQUATIONS |
|---|---|---|---|---|---|
| RESISTOR | 1-PORT | R | RESISTANCE RES IN OHMS | | V=RES×I |
| CAPACITOR | 1-PORT | C | CAPACITANCE CAP IN FARADS | | I=S×CAP×V |
| INDUCTOR | 1-PORT | L | INDUCTANCE IND IN HENRIES | IND≠0 | V=S×IND×I |
| STRAIGHT-THROUGH CONNECTION | 2-PORT | WTHRU | (NONE) | | V1=V2; I1=-I2 |
| POLARITY REVERSE | 2-PORT | WR | (NONE) | | V1=-V2; I1=I2 |
| MUTUAL INDUCTOR | 2-PORT | L | INPUT SELF-INDUCTANCE L1 IN HENRIES<br>OUTPUT SELF-INDUCTANCE L2 IN HENRIES<br>MUTUAL INDUCTANCE M IN HENRIES | M≠0 | V1=S×(L1×I1)+M×I2<br>V2=S×(M×I1)+L2×I2 |
| IDEAL TRANSFORMER | 2-PORT | IT | TURNS RATIO N | N≠0 | V1=N×V2; I1=-I2÷N |
| OPERATIONAL AMPLIFIER | 2-PORT | OPAMP | OPEN-CIRCUIT VOLTAGE GAIN A<br>OUTPUT IMPEDANCE ROUT IN OHMS<br>INPUT IMPEDANCE RIN IN OHMS | A≠0<br>RIN≠0 | V1=RIN×I1<br>V2=(A×V1)+ROUT×I2 |
| OPERATIONAL AMPLIFIER | 2-PORT | OPAMP | OPEN-CIRCUIT VOLTAGE GAIN A<br>OUTPUT IMPEDANCE ROUT IN OHMS | A≠0 | I1=0<br>V2=(A×V1)+ROUT×I2 |
| OPER. AMPLIFIER | 2-PORT | OPAMP | VOLTAGE GAIN A | A≠0 | I1=0; V2=A×V1 |
| FIELD-EFFECT TRANSISTOR MODEL, GROUNDED-SOURCE | 2-PORT | FET | GATE-SOURCE CAPACITANCE CGS IN FARADS<br>GATE-DRAIN CAPACITANCE CGD IN FARADS<br>TRANSCONDUCTANCE GM IN MHOS | GM≠0 | I1=S×(CGS×V1)+CGD×V1-V2<br>I2=(GM×V1)+S×CGD×V2-V1 |
| BIPOLAR-TRANSISTOR MODEL, GROUNDED-EMITTER | 2-PORT | HYBRIDPI | RESISTANCE RX IN OHMS<br>RESISTANCE RPI IN OHMS<br>CAPACITANCE CPI IN FARADS<br>CAPACITANCE CMU IN FARADS<br>TRANSCONDUCTANCE GM IN MHOS | RPI≠0<br>GM≠0 | V1=VPI+RX×I1<br>I1=(VPI×(S×CPI)+÷RPI)+<br>S×CMU×VPI-V2<br>I2=(GM×VPI)+S×CMU×V2-VPI |
| LOSSLESS TRANSMISSION LINE | 2-PORT | TEM | CHARACTERISTIC IMPEDANCE Z0 IN OHMS<br>LENGTH LEN IN METERS | Z0≠0 | A=*-J×O2×LEN×F÷3E8÷DIEL*.5<br>(V2-Z0×I2)=A×V1+Z0×I1<br>(V1-Z0×I1)=A×V2+Z0×I2 |
| TRANSMISSION LINE CHARACTERISTIC IMPEDANCE | 1-PORT | TEM | CHARACTERISTIC IMPEDANCE Z0 IN OHMS | Z0≠0 | V=Z0×I |
| LOSSLESS WAVEGUIDE DOMINANT MODE | 2-PORT | WG | CUTOFF FREQUENCY FC IN HERTZ<br>INFINITE-FREQUENCY CHARACTERISTIC IMPEDANCE ZINF IN OHMS<br>LENGTH LEN IN METERS | ZINF≠0<br>~FC<F | Z0=ZINF÷(1-(FC÷F)*2)*.5<br>A=J×LEN×ZINF×F÷Z0×3E8÷DIEL*.5<br>(V2-Z0×I2)=(*-O2×A)×V1+Z0×I1<br>(V1-Z0×I1)=(*-O2×A)×V2+Z0×I2 |
| WAVEGUIDE CHARACTERISTIC IMPEDANCE | 1-PORT | WG | CUTOFF FREQUENCY FC IN HERTZ<br>INF.-FREQ. CHAR. IMP. ZINF IN OHMS | ZINF≠0<br>~FC<F | V=I×ZINF÷(1-(FC÷F)*2)*.5 |

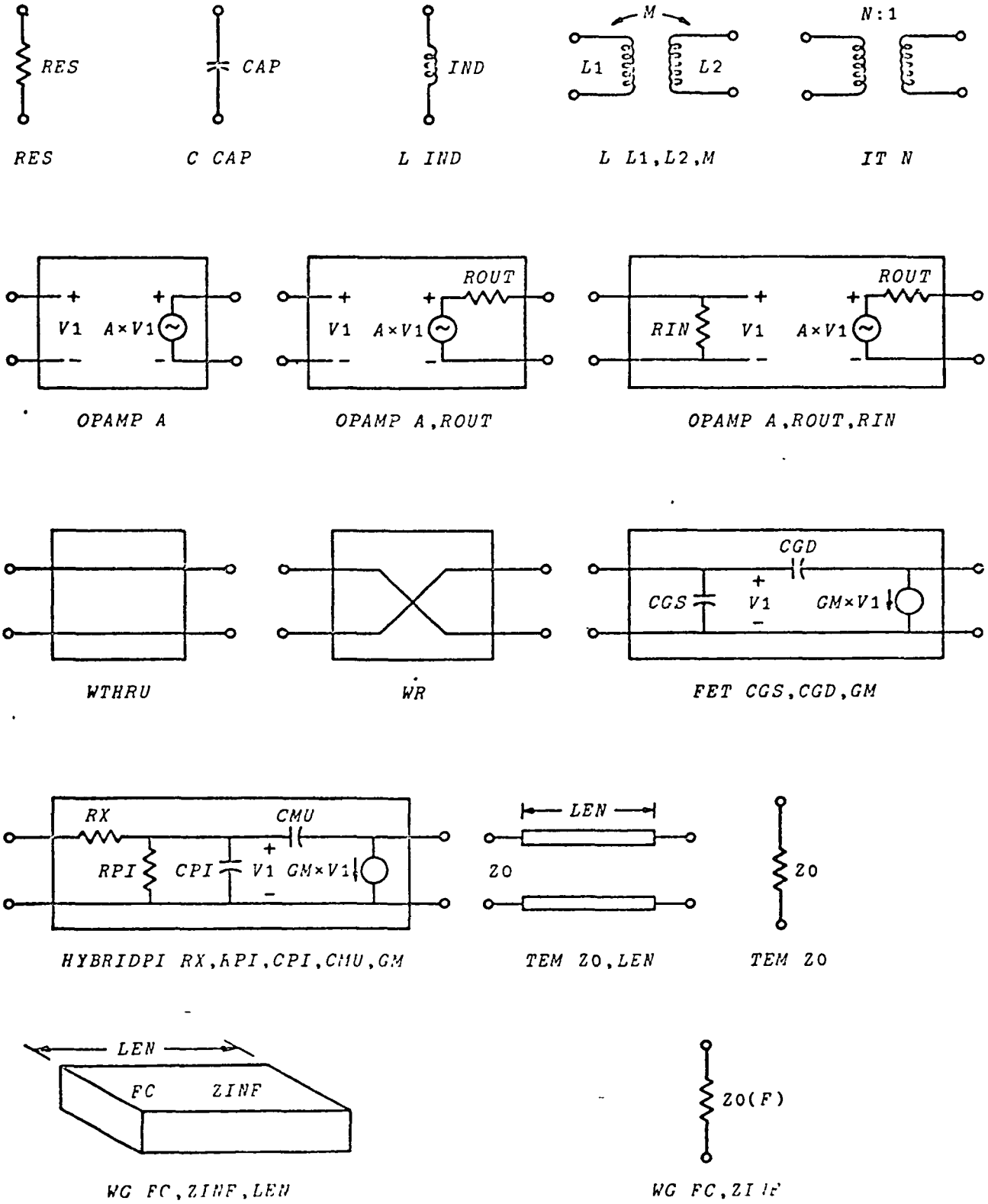



Figure 2. Elements defined in *MARTHA*.

Note the two auxiliary functions, *DEGREESAT* and *FORDIEL*. The first is used to specify an electrical length in degrees at a reference frequency (in this case 2 GHz) and the second is used for dielectrically loaded guides.

The library workspace 100 *MARTHAE* contains about 50 additional elements and models. One of them is the voltage-controlled voltage source, *VCVS*. That is a 2-port element, with the controlling branch on the input and the controlled branch on the output. Current can be used as well as voltage for either the controlling or controlled branch, so there are three additional elements, named *VCCS*, *CCVS*, and *CCCS*. In addition, if flux linkage or charge are possible as the controlling or controlled variable, there are 12 additional controlled sources, including for example a charge-controlled current source, *QCCS*. These are useful for modeling.

Another function useful for modeling is *ZPDE*. If its argument is of length 2, the first is an integer and the second is a coefficient, the result is a 1-port network whose impedance is equal to the coefficient times complex frequency s raised to the integer power. If the integer is 0, the result is a resistor; if it is 1, the result is an inductor. Values of the integer from -5 through 5 are possible. *ZPDE* also produces, if its argument is of length 3, 4, or 5, a 2-port similarly defined element where the numbers in the argument are from the 2-port impedance matrix. Other functions named

*YPDE*, *HPDE*, and *ABCDPDE* operate similarly. These are useful for modeling using a power series expansion.

Other elements in the *MARTHA* library include gyrators, nullors, negative-impedance convertors, and attenuators and isolators, both for TEM lines and waveguides. Included also are functions for converting numerically defined functions of frequency (*FOF*'s) into elements. The *FOF* can be interpreted as impedance, admittance, or reflection coefficient of a 1-port network, or as impedance, admittance, hybrid, ABCD, or scattering matrix of a 2-port network.

The workspace 100 *MARTHAE* contains more elements than can be described here. Complete documentation appears in references 1 and 2.

The library workspace 100 *MARTHAX* contains several auxiliary functions for working with *MARTHA*, many of which are useful models. Examples are functions to calculate characteristic impedance of coaxial or microstrip transmission lines, or calculate coaxial discontinuity capacitances, or calculate cutoff frequency and characteristic impedance of common waveguides.

## IV. INPUT LANGUAGE

*MARTHA* is interactive. The user sits at a computer terminal and types his input line, and gets an immediate reply.

Some of the things a user must type have nothing to do with network analysis. After the user dials the telephone number of the computer which carries *MARTHA* (which may, of course, be many miles distant) he must log in by identifying himself. He must also load *MARTHA* from the computer's public library, and perhaps copy some of the *MARTHA* library and perhaps copy some of his previous results, including previously defined networks or models, or previously calculated results. At any time the user can save his work up to that point, or start over, or log off the computer.

The rest of the user's time can be spent analyzing networks. A network can be defined at any time, and an analysis can be requested at any time. The analysis might be of a previously defined network, or of a newly defined network, possibly composed in part of previously defined networks.

The general form of an analysis request in *MARTHA* is

$$\left\{\begin{array}{l} PRINT \\ PLOT \\ PLOG \\ SMITH \\ STORE \end{array}\right\} \quad \text{<output list>} \; OF \; \text{<network>}$$

The first word used (*PRINT*, *PLOT*, *PLOG*, *SMITH*, or *STOPE*) fixes the basic format of the output; *PLOG* is used for a plot with a logarithmic scale for the independent variable. The form of the "output list" is discussed in Section V. The word *OF* is required to separate the output list from the network description.

The form of the network description is unique to *MARTHA*. Two types of APL functions are used to define networks. The first create elements, for example, resistors, capacitors, etc., and the second create networks out of other networks, by wiring them together. Element-definition functions were described in Section III. An example of a wiring function is *S*, which is dyadic and, like all dyadic APL functions, has an argument on each side. Thus, if *A* and *B* are 1-port (2-terminal) networks then *A S B* is the new 1-port network formed by putting the two networks *A* and *B* in series. Similarly, the function *P* wires two networks in parallel. Figure 1 shows a schematic diagram and a description in *MARTHA* notation of a simple series-parallel network. The functions *S* and *P* are sufficient to wire together all linear series-parallel networks. For more complicated topology, other wiring functions, designed to work on 2-port networks, are defined in *MARTHA*.

There are fourteen wiring functions in *MARTHA*, including the functions *S* and *P*. These are shown in Figure 3. To create 2-port networks out of 1-port networks, the functions *WS* and *WP* are used. These are monadic; that is, they have one argument instead of two. The argument appears on the right of the

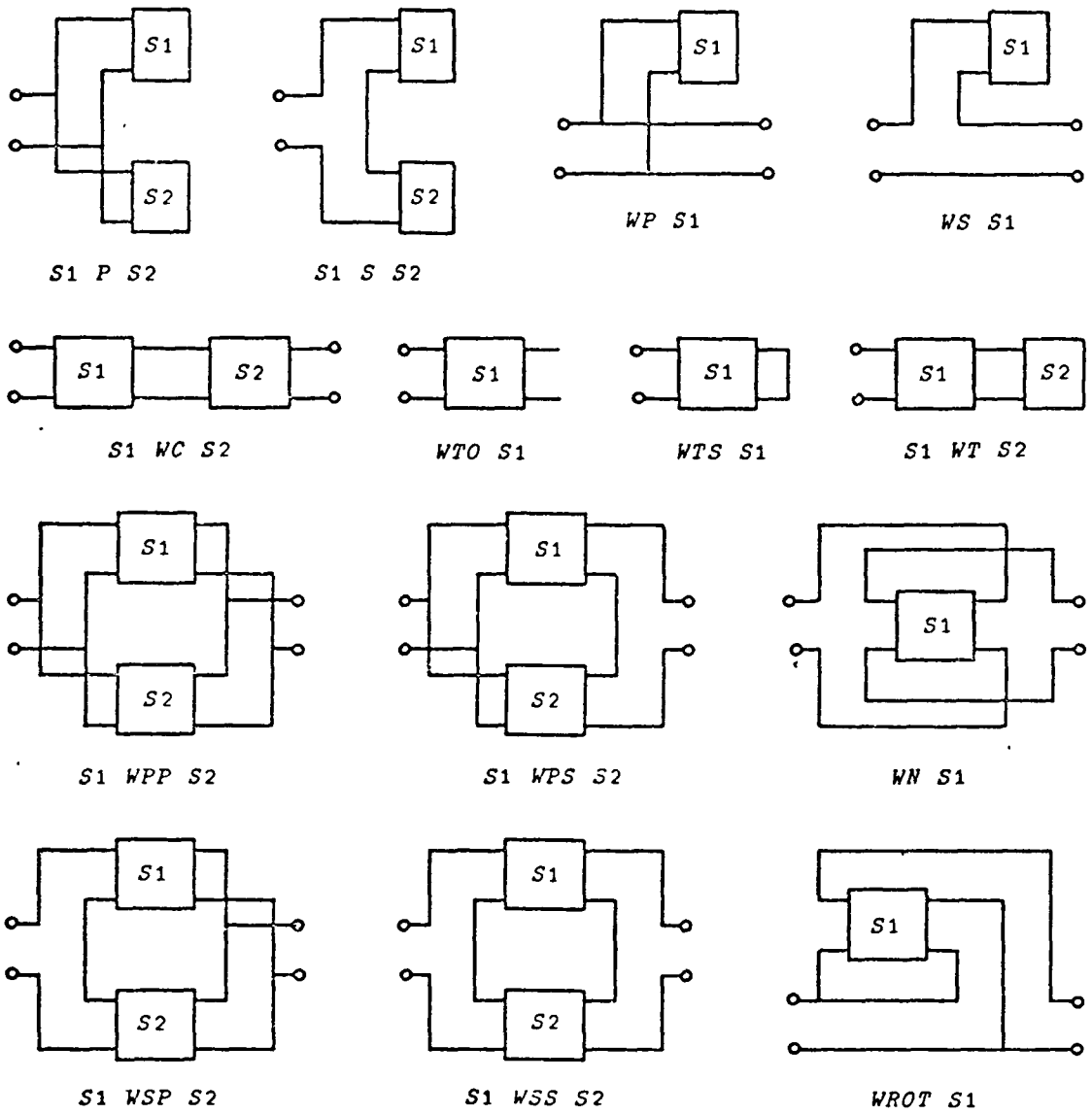



Figure 3. Wiring functions defined in *MARTHA*.

function, as is true of all monadic APL functions. Two 2-port networks can be wired together several ways. The functions *WPP*, *WPS*, *WSP*, and *WSS* connect the two inputs and the two outputs, using either parallel or series connections at each port. These are useful in denoting feedback amplifiers; for example, the function *WSS* might be used for emitter degeneration. The cascade function *WC* is very common. The monadic functions *WH* and *WROT* are useful in converting grounded-emitter transistors to grounded-base or grounded-collector configurations. Symmetrical filters can be denoted easily with *WM*. if the left-hand side of the filter is called *A*, then the overall filter is *A WC WH A*. Finally, three techniques for converting a 2-port network to a 1-port network by terminating the output port are shown. The output port can be open-circuited (*WTO*) or short-circuited (*WTS*) or terminated in another network (*WT*). Note that *WTO* and *WTS* are monadic, but *WT* is dyadic, expecting a 2-port network as its left argument and a 1-port network as its right argument.

A precedence rule for the wiring functions must be established. By that is meant a rule for determining which of the functions are to be considered executed before others. For example, in ordinary algebraic notation, the expression AxB+3, written without parentheses, indicates that the multiplication is to be performed before the addition. This is an example of the common rule that exponentiation is performed

before multiplication and division, and those are performed before additions and subtractions. By way of contrast, the language APL has a simpler precedence rule. The rule is that all functions have equal precedence, and are executed strictly in the order indicated, from right to left, unless parentheses are used to delimit expressions which are to be evaluated first. Thus, in APL, $A\times B+3$ would be $A\times(B+3)$, rather than $(A\times B)+3$. The precedence rule for the wiring functions in *MARTHA* is the same as that of APL. As a consequence, every monadic wiring function takes as its argument the entire expression to its right, and every dyadic function takes as its left argument the one object immediately to its left, and as a right argument the entire expression to the right. Parentheses can, of course, be used in the usual way to surround expressions which are to be evaluated first. Parentheses are often required for left arugrents of dyadic functions, but are never required (though they are permitted) for right arguments of functions. As an example, the expression *A1 S A2 P A3* refers to the network of Figure 4(a), rather than the network of Figure 4(b), which would be written *(A1 S A2) P A3*.

Some of the wiring functions expect 1-port networks and others 2-port networks as arguments. What happens if the wrong kind of network is used? For example, what happens in *A S B* if *B* is a 2-port network? In order to allow all wiring functions to operate on both 1-port and 2-port networks, two "automatic conversion conventions" are adopted in MARTHA. They are:

1. If a wiring function expects a 1-port network and encounters a 2-port network then the output is open-circuited and the input is used. This is equivalent to using the function *1TO*.

2. If a wiring function expects a 2-port network and encounters a 1-port network, then the wiring function *2P* is automatically invoked to convert to a 2-port network.

Several examples of networks are shown both in *MARTHA* notation and in schematic diagrams, in Figure 5. These include the widely-used Darlington transistor connection, a half-ladder and a Wheatstone bridge.

*MARTHA* incorporates several other functions that are not, strictly speaking, wiring functions, although they have as an argument a network and return as a result a network based upon the argument. All of these operate on both 1-port and 2-port networks, and return a network with the same number of ports. The monadic function *ADJ* converts a network to its adjoint[6-8], which is the network whose impedance and admittance matrices are the transposes of the corresponding matrices for the original network. The function *ZSCALE* is dyadic; its left argument is a number and the result is a network with the same topology as the network of its right argument, but with all elements scaled in impedance. Similarly, *FSCALE* performs a frequency scaling. These functions are useful in filter designs, where perhaps a 1-ohm, 1-Hz prototype is known. In a similar way, *HINVERT* performs a low-pass to high-pass transformation, and *FBP* a low-pass to band-pass transformation.

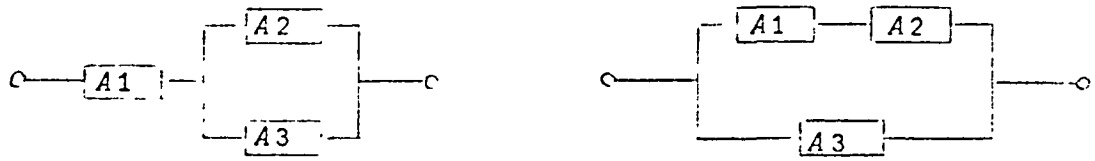
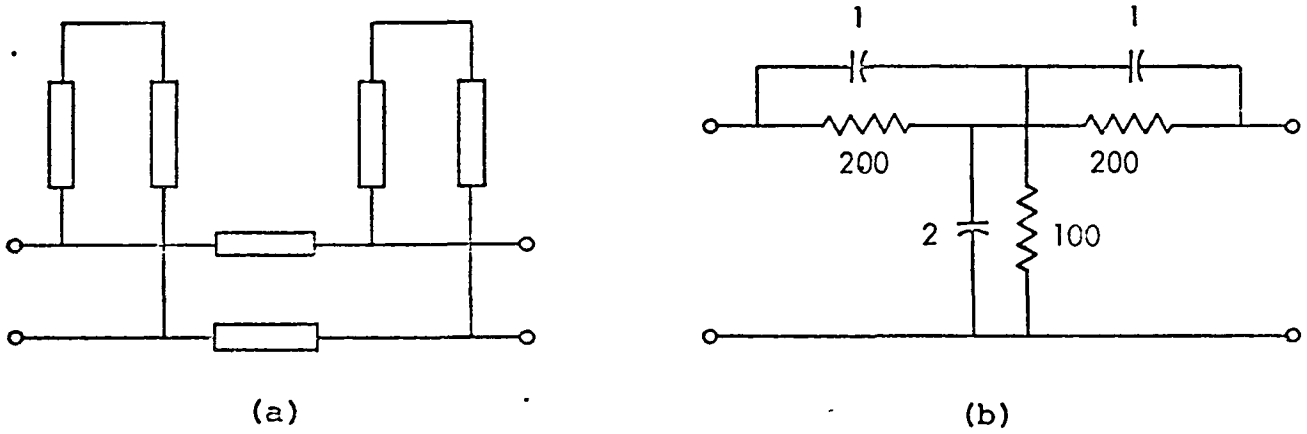



Figure 4. Illustration of the precedence convention for wiring functions in *MARTHA*.



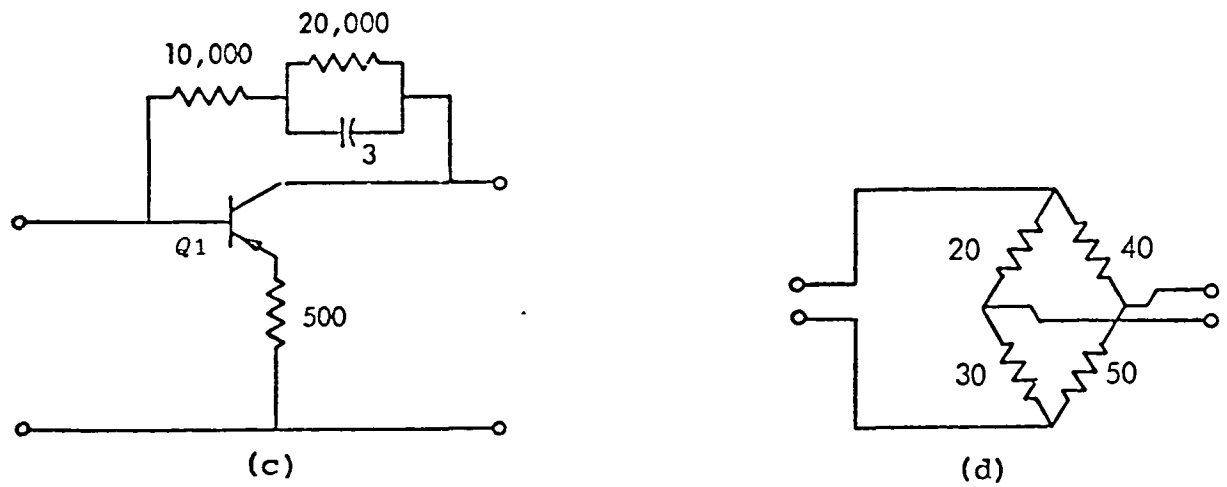




Figure 5 (continued on next page)

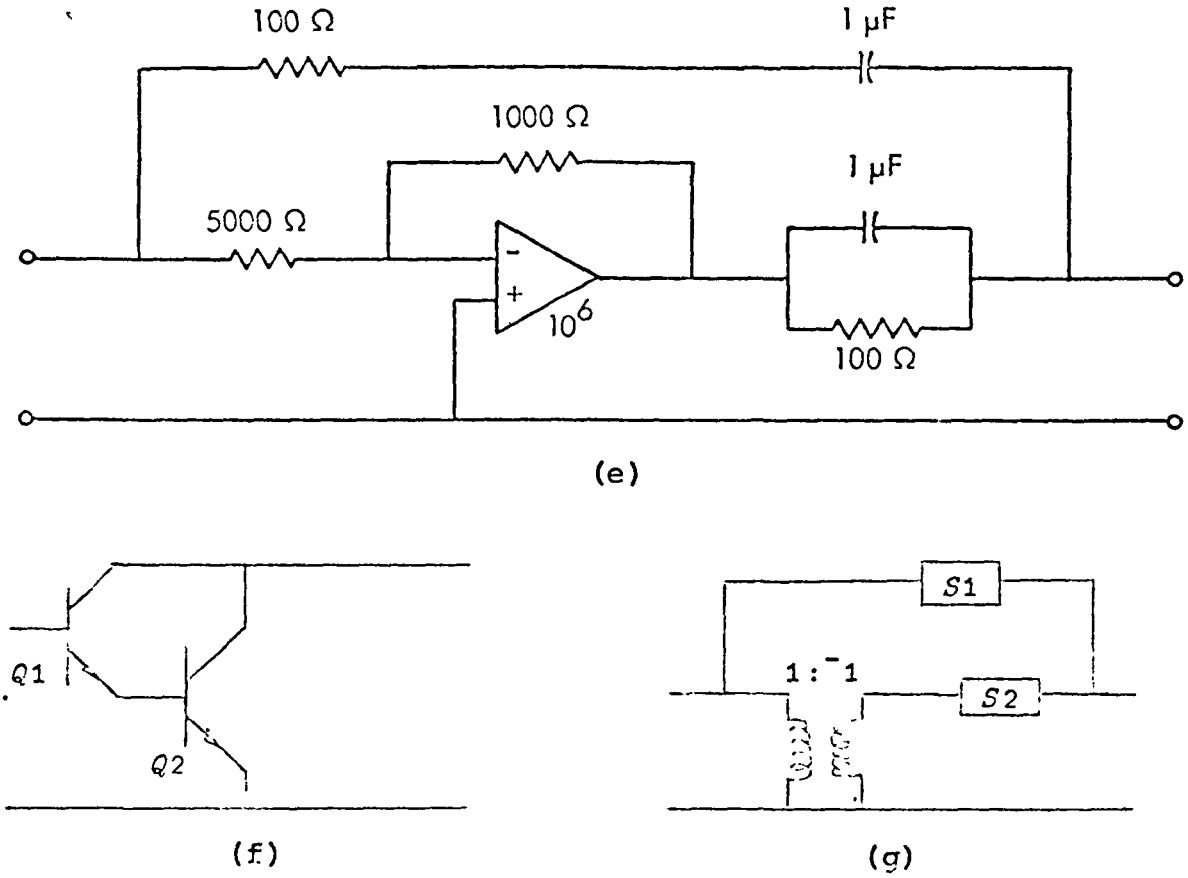



Figure 5. Examples of *MARTHA* notation.

(a) Double-stub tuner with 50-ohm lines and adjustable line lengths *L1* and *L2*, and fixed separation 5 inches:

```
(WTS TEM 50,L1) WC (TEM 50,5×.0254) WC WTS TEM 50,L2
```

(b) Twin-tee filter, values in ohms and microfarads:

```
((WS R 200)WC(C 2E⁻6)WC WS R 200)WPP(WS C 1E⁻6)WC(R 100)WC WS C 1E⁻6
```

(c) Feedback amplifier:

```
(Q1 WSS R 500) WPP WS (P 10E3) S (R 20E3) P C 3E⁻6
```

(d) Wheatstone bridge:

```
((WS R 40) WC R 50) WPS WR WC (WS R 20) WC R 30
```

(e) Active allpass filter (reference 5):

```
(WS(R 100)S C 1E⁻6)WPP(WS R 5000)WC((OPAMP ⁻1E6)WPP WS R 1000)WC
      WS(R 100)P C 1E⁻6
```

(f) Darlington transistor connection:

```
WROT WROT (WROT Q2) WC WROT Q1
```

(g) Half-lattice network:

```
(WS S1) WPP (IT ⁻1) WC WS S2
```

The latter two may be used in succession to define a band-elimination filter. The function *WCC* returns a network with every element replaced by its "complex conjugate"; it is useful in defining conjugate-match loads for optimum power transfer. Finally, the function *NDUAL* takes the dual of a network, changing topology, and element values, and element types in the process.

This last set of functions is useful when *MARTHA* is used for design work. The *MARTHA* library contains a function named *WHATIS* which prints network descriptions of any network. Thus , for example, if *PROTO* is a 1-ohm, 1-Hz prototype filter, then

```
DESIGN ← 1000 ZSCALE 1E6 FSCALE PROTO
```

defines a 1000-ohm, 1-MHz filter. Then

```
WHATIS DESIGN
```

will print its definition, and

```
PLOT (DB IG OF DESIGN), DB IG OF DESIGN WC WS R 800
```

will analyze it (in one case after it is wired with another resistor).

# V. OUTPUT SPECIFICATIONS

The general form of an analysis request in *MARTHA* is

$$\left\{\begin{matrix} PPINT \\ PLOT \\ PLOG \\ SMITH \\ STORE \end{matrix}\right\} \quad \text{<output list> } OF \text{ <network>}$$

In this section the output list is described.

The output list is, basically, a list of response functions of the network to be calculated. It also can include modifiers on the response functions, and some format requests.

## A. Response Functions

*MARTHA* can calculate over a hundred different response functions, of which the thirty in Table 2 are thought to be of wide interest, and the remainder of specialized interest. The response functions of Table 2 are in *MARTHA* , and the rest in the library workspace 100 *MARTHAR*.

For 1-port networks, the response functions are the impedance *Z*, admittance *Y*, normalized impedance $\underline{Z}$ and admittance $\underline{Y}$, reflection coefficient *SC*, and voltage standing-wave ratio *VSWR*. Of these, *VSWR* is real and the others are complex. If the network in question is a 2-port network, then (in accordance with the automatic conversion convention for wiring functions), its output port is open-circuited and the input port is used.

Table 2. Response functions in *MARTHA*. Of these 30, 26 are complex and four real. Over 70 additional response functions are in the library workspace 100 *MARTHAR*.

| NAME | C/R | MEANING | DEPENDS ON |
|---|---|---|---|
| Z | C | IMPEDANCE OF A 1-PORT NETWORK<br>V=Z×I | NETWORK |
| Y | C | ADMITTANCE OF A 1-PORT NETWORK<br>I=Y×V | NETWORK |
| SC | C | REFLECTION COEFFICIENT OF 1-PORT<br>B=SC×A | NETWORK, ZO |
| Z11 | C | IMPEDANCE MATRIX<br>V1=(Z11×I1)+(Z12×I2) | NETWORK |
| Z12 | C | | |
| Z21 | C | V2=(Z21×I1)+(Z22×I2) | |
| Z22 | C | | |
| Y11 | C | ADMITTANCE MATRIX<br>I1=(Y11×V1)+(Y12×V2) | NETWORK |
| Y12 | C | | |
| Y21 | C | I2=(Y21×V1)+(Y22×V2) | |
| Y22 | C | | |
| H11 | C | HYBRID MATRIX<br>V1=(H11×I1)+(H12×V2) | NETWORK |
| H12 | C | | |
| H21 | C | I2=(H21×I1)+(H22×V2) | |
| H22 | C | | |
| S11 | C | SCATTERING MATRIX<br>B1=(S11×A1)+(S12×A2) | NETWORK, ZIN, ZOUT |
| S12 | C | | |
| S21 | C | B2=(S21×A1)+(S22×A2) | |
| S22 | C | | |
| ZIN | C | INPUT IMPEDANCE V1÷I1 | NETWORK, ZL |
| YIN | C | INPUT ADMITTANCE I1÷V1 | NETWORK, ZL |
| SIN | C | INPUT REFLECTION COEFFICIENT B1÷A1 | NETWORK, ZL, ZIN |
| ZOUT | C | OUTPUT IMPEDANCE<br>V2÷I2 WHEN OUTPUT EXCITED | NETWORK, ZG |
| YOUT | C | OUTPUT ADMITTANCE<br>I2÷V2 WHEN OUTPUT EXCITED | NETWORK, ZG |
| SOUT | C | OUTPUT REFLECTION COEFFICIENT<br>B2÷A2 WHEN OUTPUT EXCITED | NETWORK, ZG, ZOUT |
| VG | C | VOLTAGE GAIN V2÷EG | NETWORK, ZG, ZL |
| AG | R | AVAILABLE GAIN POUT,AV÷PIN,AV | NETWORK, ZG |
| IG | R | INSERTION GAIN<br>POUT(NETWORK)÷POUT(WTHRU) | NETWORK, ZG, ZL |
| PG | R | POWER GAIN POUT÷PIN | NETWORK, ZL |
| TG | R | TRANSDUCER GAIN POUT÷PIN,AV | NETWORK, ZG, ZL |

In calculating *SC*, *VSWR*, *Z*, and *Y* a normalization impedance is required; in *MARTHA* the normalization impedance for 1-port networks is known as *ZN*, and must be set by the user before analysis. It may be either an impedance (a real number) or a real or complex nurerical function of frequency (*FOF*) or any 1-port network defined in *MARTHA* notation (in which case the impedance of the network will be used).

For 2-port networks, there is a wide variety of response functions. If a 2-port response function is requested of a 1-port network, then (in accordance with the automatic conversion convention for wiring functions) the wiring function *WP* is assumed to be invoked. The response functions available include all the common 2-port parameters (impedance, admittance, hybrid, ABCD matrix, and others) and the corresponding matrices for wave variables (including the scattering matrix and scattering transmission matrix). For calculations of the wave-variable responses, normalization impedances at both the input and the output are required; these are known as *ZNIN* and *ZNOUT*, and may be different both from each other and from *ZN*. Each of these, like *ZN*, may be a real constant, real or complex numerically defined function, or any *MARTHA* 1-port network.

Many of the 2-port response functions depend also on the generator and/or load impedance. A 2-port network is assumed to be terminated by generator and load as shown in Figure 6. the variables *ZG*, *ZL*, and *EG* may, like *ZN*, be specified in

several different ways, but they must be specified before analysis. Examples of response functions include the input impedance, admittance, reflection coefficient, and VSWR (these all depend on *ZL*) and corresponding output quantities (these all depend on *ZG*). Other examples are various measures of gain, including power gain, voltage gain, open-circuit voltage gain, voltage ratio, transducer gain, insertion gain, insertion voltage gain, unilateral gain, conjugate-match gain, and available gain. Also available are various characteristic impedances of the network, including image impedance, iteration impedance, and conjugate-match impedance, along with corresponding admittances, reflection coefficients, and propagation constants. Also included are stability factors for amplifiers, and normalized impedances and admittances. Other response functions include the input and output voltages, currents, and wave variables.

## B. Modifiers

The complex response functions appear normally in the form of real and imaginary parts. This may be changed by the use of modifiers. Table 3 lists the most important modifiers in *MARTHA*. For complex responses, the real part, imaginary part, magnitude, magnitude in decibels, angle, and phase delay are all useful. The appropriate modifiers are placed before

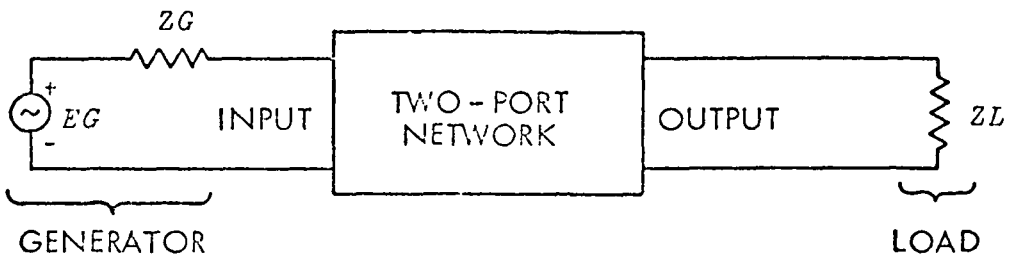



Figure 6. Termination of two-port networks assumed when some of the response functions are calculated. The generator and load impedances *ZG* and *ZL* are not part of the network definition.

Table 3. Modifiers in *MARTHA*. Of thses six, *RE*, *IM*, *RAD*, *PD*, and *DEG* are ignored if applied to a real response, and *DB* is 20×10⌾*MAGNITUDE* for complex responses and 10×10⌾*ABSOLUTE VALUE* for real responses. If no modifier is used, the real and imaginary parts of complex responses will result. Other modifiers are in the library workspace 100 *MARTHAM*.

| MODIFIER | MEANING |
|---|---|
| *RE* | Real Part |
| *IM* | Imaginary Part |
| *MAG* | Magnitude |
| *RAD* | Phase in radians |
| *DEG* | Phase in degrees |
| *DB* | Magnitude in dB. |
| *PD* | Phase delay |
| *REC* | Reciprocal |

the responses in question. Example:

*PRINT MAG Z, DEG Z OF (R 1) P L .02*

Each of the modifiers acts on only the one response immediately to its right, contrary to the normal APL convention that functions have as their right argument the entire expression to their right. Thus parentheses are not ordinarily used in the output list.

C. Formats

Various format requests can also be inserted in the output lists, generally at any point in the list. For prints, the number of significant figures printed can be changed from its normal value of 5 by the function *PLACES*. Example:

*PRINT 7 PLACES DB IG OF CRYSTALFILTER*

For plots, several additional options are available. Plots are normally made against frequency as the independent variable, but if desired any one of the responses will act as the independent variable if it is preceded in the output list by *VS*. Alternatively, if *PAIRS* appears in the output list, the first response function will be plotted against the second, and the third against the fourth, etc. Plots are normally 50 spaces wide and 50 lines high, to fit conveniently on one page (except those made by the function *SMITH* which are designed to fit a standard-size Smith chart).

Other widths and heights can be specified in the output list. Plotting characters other than the standard ones can be specified by the function *SYMBOLS*. Normally *MARTHA* plots all dependent variables with different scales, selecting each so that the scales consist of round numbers, but still significant detail is shown in each plot. The dependent variables will all have the same scale (with usually some loss of detail) if *SS* appears in the output list. The horizontal and vertical scales can be set to arbitrary values by the functions *HSCALE* and *VSCALE*. This is useful in magnifying certain critical regions of the plot; points falling outside the specified scales are simply ignored.

These plotting format functions are illustrated by some examples:

```
PLOG 'MD' SYMBOLS MAG Z, DEG Z OF NETWORK

PLOT SS Z11, Z12, Z22 OF FILTER

PLOT -20 20 HSCALE -180 180 VSCALE DB RR VS DEG RR
   OF AMPLIFIER

SMITH S11, SIN, S12 OF FILTER

PLOT PAIRS (Z OF AMP1), ZIN, Z11 OF AMP2
```

The third request prints the Nichols chart for the return ratio *RR* of the amplifier, and the fourth request produces a standard-size Smith chart.

# VI. MODELS

A computation using *MARTHA* (or any other program) will only be accurate if reasonable care is used in modelling the network. Devising suitable models for a network is usually the most challenging part of any analysis. *MARTHA* cannot, of course, do the modelling job for the user, but it does offer him a selection of elements useful for models, the possibility of numerically defined elements, and the option of creating other elements in the form of "user-defined elements".

## A. Types of Models Allowed

Models in *MARTHA* may be any network with topology definable in the *MARTHA* notation, using any of the built-in *MARTHA* elements, along with any linear 1-port or 2-port element not already in *MARTHA*, for which the user is able to supply an APL algorithm for computing its impedance (for 1-port elements) or its ABCD matrix (for 2-port elements).

Note that *MARTHA* does not distinguish models from elements, or from networks containing several elements wired together. If any user finds that he uses a given configuration frequently, then he can write a simple APL function that returns that particular network upon demand. For users who wish a model that is not representable by a network made up from built-in *MARTHA* elements, *MARTHA* allows user-defined elements without restriction as to complexity of the required algorithm.

Several elements of special interest in modelling were discussed in Section III. *MARTHA* can also handle numerically defined elements. To create one of these, the user types in a table of values and then calls one of the functions *ZFOF*, *YFOF*, *SFOF*, *HFOF*, and *ABCDFOF* to interpret the table as numerical values of impedance, admittance, or reflection coefficient for 1-port networks, or Z, Y, S, H, or ABCD matrices for 2-port networks. The resulting numerical models are then treated by *MARTHA* like other elements. Arbitrary frequency dependence is available this way.

## B. Built-In Models

*MARTHA* contains several built-in models, of which the most important are probably the hybrid-pi and FET transistor models, and four simple models of operational amplifiers. These are considered as elements in *MARTHA* , and were discussed in Section III.

## C. Input Techniques and Format

For user-defined models, a distinction should be made between those models that are networks of *MARTHA* elements, and those that are not. For the former, a simple APL function with an argument will usually suffice. As an example, consider the Marcuvitz model[9] for a window formed from a pair of semicircular obstacles along the sides of a waveguide, Figure 7.

The equivalent circuit shown has inductors with inductances

$$L_A = \frac{Z_\infty}{2\pi f_c} \left(\frac{c}{2\pi f_c D}\right)^2$$

$$L_B = -\frac{Z_\infty}{16\pi f_c} \left(\frac{2\pi f_c D}{c}\right)^4$$

Note that $L_B$ is negative. $Z_\infty$ is the characteristic impedance of the waveguide at infinite frequency, $f_c$ is the cutoff frequency and c the speed of light. A function that returns this 2-port network is

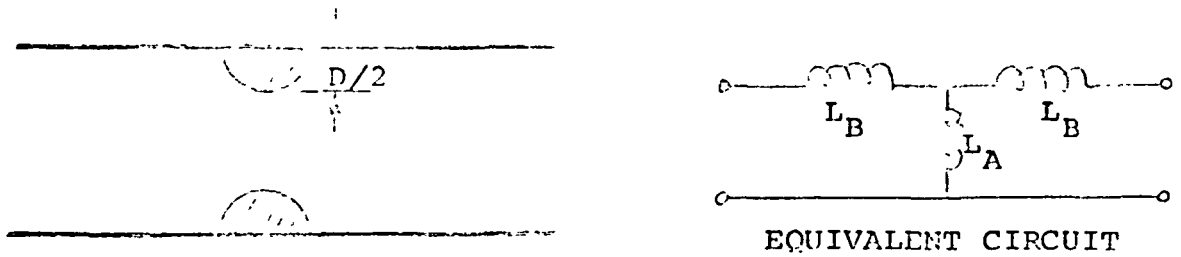
```
     ∇ B←WINDOW A
[1]    X←300000000÷(6.28×A[1]×A[3])
[2]    LA←A[2]×(X*2)÷(6.28×A[1])
[3]    LB←-A[2]÷((X*4)×16×3.14×A[1])
[4]    B←(WS L LB) WC(L LA) WC WS L LB
     ∇
```

where the argument *A* is a vector of length 3 containing the cutoff frequency, the impedance, and the diameter. A relatively small amount of APL programming ability is required to read or write this model. The normal APL function-definition scheme is used, and all APL primitive functions (and *MARTHA* functions) may be used.

Next, consider the case where the model is so complicated that a network with *MARTHA* elements is not sufficient to describe it. The technique now is to use *MARTHA*'s user-defined-element capability. Two kinds of functions are required, the first to define the user-defined element, and the second to actually compute its response at each frequency. The first function is easy to write: it should return an APL vector of any length starting

Table 4. Required shape and interpretation of matrices returned by the user-written function *NEWELEMENT*.

| | Number of Columns | | |
|---|---|---|---|
| Column | 2 | 6 | 8 |
| 1 | Re Z | Re A | Re A |
| 2 | Im Z | Im A | Im A |
| 3 | | Re B | Re B |
| 4 | | Im B | Im B |
| 5 | | Re C | Pe C |
| 6 | | Im C | Im C |
| 7 | | | Re D |
| 8 | | | Im D |
| Assumed | | $D = \frac{(1+BC)}{A}$ | |
| Properties of Result | 1-Port | 2-Port Reciprocal | 2-Port |




TOP VIEW EQUIVALENT CIRCUIT

Figure 7. Symmetrical window between two semicircular inductive posts in a waveguide and the model given by Marcuvitz[9].

with the number 9. The rest of the numbers in the vector contain the parameters associated with the model. The second function is more difficult. It must be a monadic APL function entitled *NEWELEMENT*. It is not called directly the the user, but instead will be called by one of the *MARTHA* functions when analysis is done. At that time the frequency is known (it need not be known during the definition of the network) and *NEWELEMENT* may refer to it. The function *NEWELEMENT* must return either a 2-dimensional matrix according to Table 4, or a *MARTHA* network, possibly incorporating such a matrix as an element. The first dimension of the matrix is in each case the number of frequencies, and the second dimension is 2, 6 or 8, depending on whether the network is a 1-port network, a reciprocal 2-port network, or a nonreciprocal 2-port network.

As an example, consider a length of coaxial transmission line with skin-effect loss. The TEM lines in *MARTHA* are lossless, and the frequency dependence of the skin-effect is not given exactly by any *MARTHA* element. For this example, the necessary parameters to specify the line are the inner and outer radii, and the length. A function that creates the user-defined element is very simple:

```
     ∇ B←COPPERLINE A
[1]    B← 9 2.52E¯7 ,A
     ∇
```

The argument *A* is assumed to consist of the inner and outer radii and the length, all in meters. This function merely defines the

element; it does not do any calculations.

During MARTHA analysis, the function *NEWELEMENT* is called when this element is encountered. The real calculations are performed by that function. The ABCD matrix for this line is given by

$$A = D = \cosh \gamma \ell$$

$$B = Z_o \sinh \gamma \ell$$

$$C = (\sinh \gamma \ell)$$

where $\ell$ is the length and

$$\gamma = \sqrt{(R + j\omega L)(G + j\omega C)}$$

$$Z_o = \sqrt{\frac{R+j\omega L}{G+j\omega C}}$$

and where the per-unit-length quantities L, C, R, and G are given by[10]

$$L = \frac{\mu_o}{2\pi} \ln \frac{r_o}{r_i}$$

$$C = \frac{2\pi\varepsilon_o}{\ln r_o/r_i}$$

$$G = 0$$

$$R = 2.52 \times 10^{-7} \sqrt{f} \frac{1}{2\pi} \left(\frac{1}{r_o} + \frac{1}{r_i}\right)$$

A function *NEWELEMENT* that incorporates these formulas is

```
    ∇ B←NEWELEMENT A;CC;LL;RR;R0;X0;ALPHAL;BETAL
[1]   CC←6.28×8.854E¯12÷⊕A[4]÷A[3]
[2]   LL←2E¯7×⊕A[4]÷A[3]
[3]   RR←A[2]×(,F*0.5)×((÷A[4])+÷A[3])÷6.28
[4]   R0←(((LL÷CC)+(((LL÷CC)*2)+(RR÷CC×6.28×,F)*2)*0.5)÷2)*0.5
[5]   X0←-RR÷2×R0×CC×6.28×,F
[6]   ALPHAL←A[5]×RR÷2×R0
[7]   BETAL←A[5]×R0×CC×6.28×,F
[8]   B←((⍴,F),6)⍴0
[9]   B[;1 2]←⍉(2 1 ∘.○BETAL)× 6 5 ∘.○ALPHAL
[10]  B[;3 4]←⍉(2 1 ∘.○BETAL)× 5 6 ∘.○ALPHAL
[11]  B[;5 6]←(B[;3 4]×R0,[1.5] R0)+B[;4 3]×X0,[1.5]-X0
[12]  B[;5]←B[;5]÷(X0*2)+R0*2
[13]  B[;6]←B[;6]÷(X0*2)+R0*2
[14]  B[;3 4]←(B[;3 4]×R0,[1.5] R0)-B[;4 3]×X0,[1.5]-X0
    ∇
```

This function is not trivial, and some familiarity with *APL* is necessary either to read or write it.

The number 2.52E¯7 in the function *COPPERLINE* is a measure of the resistivity of copper, and is used in line [3] of *NEWELEMENT*. By including it as part of the element definition, other element-definition functions can use the same function *NEWELEMENT* for other materials, for example

```
    ∇ B←BRASSLINE A
[1]   B← 9 5.01E¯7 ,A
    ∇
```

An important aspect of every model is its range of validity. This includes permissible values for both parameters associated with the model, as well as allowed frequency ranges. Users are advised to have their model-making functions check the parameter values to see that they are reasonable. To check on frequency range, *MARTHA* can be made to print a warning whenever one or more

frequencies is outside the permissible range. To set the permissible range, the network is operated on by the function *FLIMITS*, which is a dyadic APL function with left argument a pair of frequency values limiting the allowed range. For example, the model for waveguide window given above is only valid (according to Marcuvitz[9]) between $f_c$ and $3f_c$. If the function *WINDOW* had the following fifth line

```
[5]   B←(A[1],3×A[1])FLIMITS B
```

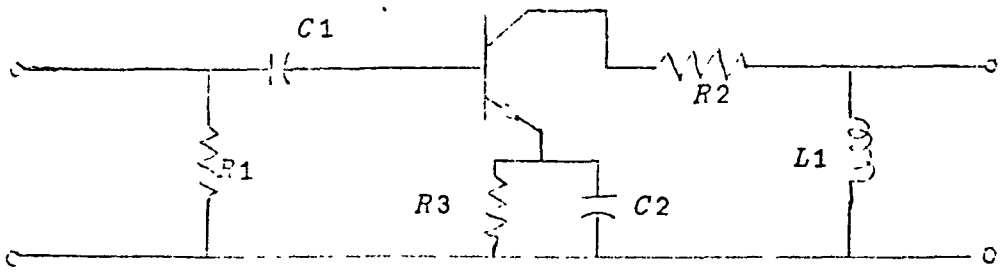
then whenever that network is analyzed, if any of the frequencies is outside that range, that fact will be reported, but analysis will continue.

# VII. EXAMPLES

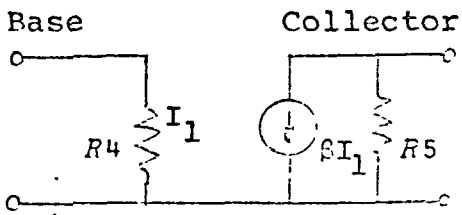
Three fully worked examples are given in this section. The first is a transistor amplifier which is also treated in other chapters of this book. The second is a crystal filter, and the third a coaxial low-pass filter. The lines typed by the user are identified by being indented six spaces. The computer response generally is not indented.

## A. Transistor Amplifier

Figure 8 shows the circuit diagram for the amplifier, together with the model to be used for the transistor. The bias source has been omitted. Because of the versatility of the technique of defining networks in *MARTHA*, the amplifier may be defined in several ways. One way which is probably appropriate for this network is suggested by Figure 9, where the input and output coupling networks, and the emitter circuit, are shown as separate two-port networks. Those circuits are straightforward, and are defined first (see *EXAMPLE* 1). The hybrid-pi transistor model used in *MARTHA* is generally regarded as the best linear model at high frequencies. This model (see Figure 2) can be made to coincide with the transistor model to be used, if *PX*, *CPI*, and *CMU* are all set to zero, and *GM* is *BETA*÷*RPI*. Then the output resistor *R5* must be added separately. The transistor is defined using this approach, and




(a)




(b)

| | | | |
|---|---|---|---|
| R1 | 50 ohms | β | 98 |
| R2 | 20,000 ohms | C1 | 10.7 pF |
| R3 | 325 ohms | C2 | 10 μF |
| R4 | 25 ohms | L1 | 1.7 mH |
| R5 | 2 Megohms | | |

(c)

Figure 8. (a) Transistor amplifier. (b) model for transistor. (c) component values.

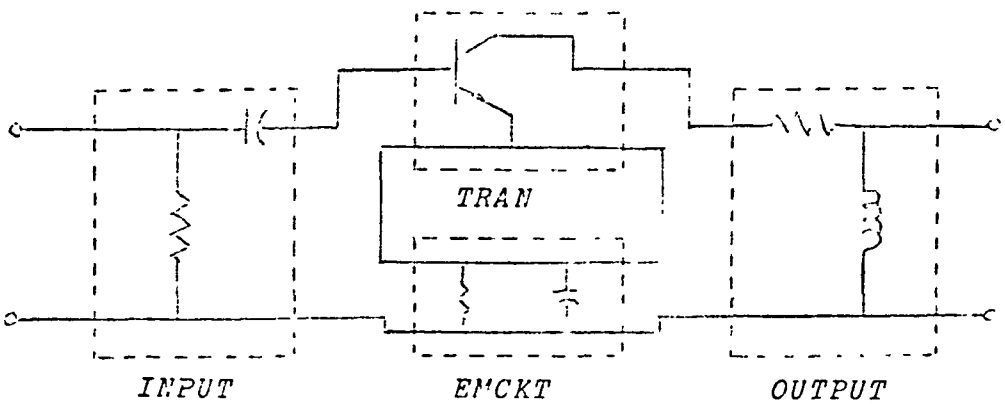



Figure 9. Transistor amplifier shown as four subnetworks wired together.

note that it is not necessary to separately calculate *BETA÷RPI*, since the formula can be typed in and the calculated result used as part of the argument for the function *HYBRIDPI*.

Refer now to *EXAMPLE 1*. First the workspace containing *MARTHA* is requested from the APL public library. Then, the four subnetworks are defined, and then wired together and given the name *AMP1*. This completes the network definition.

Before analysis, the frequency vector *F* must be specified. A study of Figure 8 reveals no mechanism for limiting high-frequency response (this is perhaps an unrealistic circuit). To view as wide a frequency range as is necessary, a logarithmic frequency sweep is used. The frequency vector *F* is set to 10 raised to the power (the asterisk is used in APL for power) of a vector ranging from 0 to 11; this covers all interesting ranges.

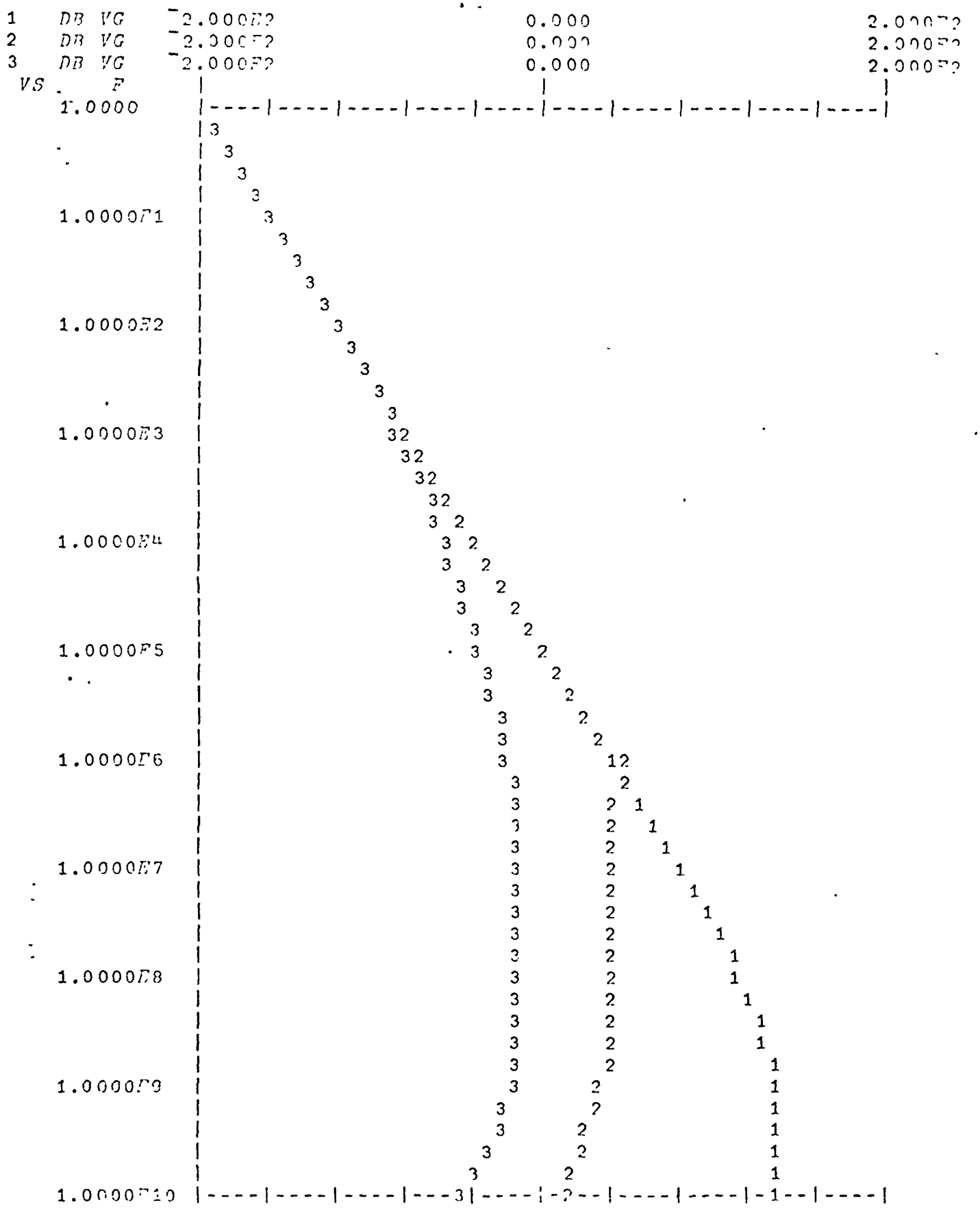
Next, a choice of response function must be made. In this example, the voltage gain *VG* is most logical, and the magnitude (in dB) and phase (in degrees) are requested. The output request begins with the word *PRINT* to set the basic format, and this is followed by the output list, then the word *OF* and finally the network definition. As expected, the voltage gain saturates at high frequencies at the rather high value of 138 dB, and the phase approaches -180 degrees.

These results must be taken with a grain of salt. Like all other network-analysis programs, *MARTHA* can give reasonable

```
      A BEGINNING OF EXAMPLE 1.

      )LOAD 100 MARNA
SAVED  14.32.54 04/23/73

      INPUT←(P 50) MC MS C 10.7⁻12
      FUNCT←(P 325) P C 10E⁻6
      OUTPUT←(MS P 2073) MC L 1.7⁻3
      TRAN←(HYBRIDPI 0, 25, 0, 0, 98÷25) MC R 2E6

      AMP1←INPUT MC (TRAN MSS FUNCT) MC OUTPUT

      F←10*0 1 2 3 4 5 6 7 8 9 10 11

      PRINT DB VG, DEG VG OF AMP1

CIRCUIT ANALYSIS BY MARNA.   73.4   5/1/73  12:22
MARNA COPYRIGHT (C) 1972 MASSACHUSETTS INSTITUTE OF TECHNOLOGY

     F          DB VG       DEG VG
---------- ----------- -----------
 1.0000     ⁻2.0314E2    6.7017E⁻5
 1.000071   ⁻1.6314E2    6.4350E⁻4
 1.000072   ⁻1.2314E2    1.2648E⁻3
 1.000073   ⁻8.3139E1   ⁻2.3785E⁻4
 1.000074   ⁻4.3139E1   ⁻3.9761E⁻3

 1.000075   ⁻3.1390     ⁻3.9921E⁻2
 1.000076    3.6861E1   ⁻3.9923E⁻1
 1.000077    7.6848E1   ⁻3.9894
 1.000078    1.1567E2   ⁻3.7407E1
 1.000079    1.3642E2   ⁻1.3254E2

 1.0000710   1.3737E2   ⁻1.7551E2
 1.0000711   1.3739E2   ⁻1.7955E2

      AMP2←AMP1 MC MP C 10E⁻12

      TRAN3←(HYBRIDPI 5, 25, 1000E⁻12, 50E⁻12, 98÷25) MC R 2E6
      AMP3←INPUT MC (TRAN3 MSS FUNCT) MC OUTPUT
```

PLOT 55 '123' SYMBOLS (DB VG OF AMP1),(DB VG OF AMP2),DB VG OF AMP3

CIRCUIT ANALYSIS BY MARTY A. 73*A 5/1/73 12:25




A END OF EXAMPLE 1.

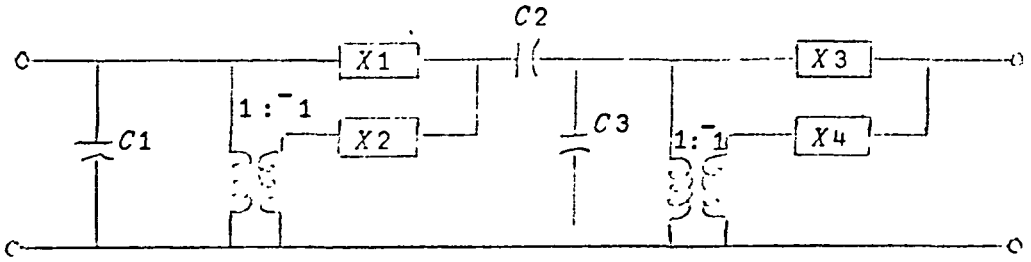
answers only if reasonable models are used, and in the present case the circuit is probably not well modelled above 1 MHz. This fact does not destroy the usefulness of the circuit as an example of how to use *MARTHA*, but it does imply the need for better models.

As an illustration of the use of *MARTHA* in judging different models, some rather simple improvements are made. First, a stray capacitance is placed across the inductor *L1*, and the resulting network named *AMP2*. This is, of course, still a crude model. Second, the transistor is re-defined with what might be typical values of *RX*, *CMU*, and *CPI*. The new network is called *AMP3*. In each case, the network definition is easy because the previously defined subnetworks are used.

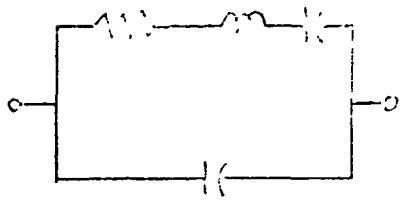
The voltage gains in the three cases are compared by simultaneously analyzing and plotting the gains of the three networks. To cover such a wide frequency range, a logarithmic plot is necessary, so the function *PLOG* is used (the function *PLOT* produces a linear plot). Note that *MARTHA* normally automatically selects the scales, but is here requested to use the same scale for the three different responses.

B. Crystal Filter

This crystal filter, Figure 10, is adapted from a filter shown to me privately by Mr. William B. Lurie, who ascribed the design to Prof. G. Szentirmai. There are two stages, each with a half-lattice construction which is modelled with the aid of an ideal transformer. The four crystals all have the same equivalent circuit, with different element values. The filter has a passband about 4000 Hz wide, between 8 MHz and 8.004 MHz. It is designed to operate between source and load of 500 ohms. Refer to *EXAMPLE 2*. First *MARTHA* is loaded, and then the four crystals are defined. Next the two stages and the overall filter are defined. Next, the generator and load impedances *ZG* and *ZL* are set to 500 ohms, and the frequency vector set so as to encompass the passband. In specifying the frequency vector, the index generator ι was used. A plot 30 lines high of the insertion gain *IG* expressed in dB, and the phase of the voltage gain in degrees is requested. Next, to see the ripple in the passband a little more clearly, the horizontal scale is deliberately set to the range from -2 to 0 dB, with the aid of the function *HSCALE* from the *MARTHA* library. This request overrides the normal *MARTHA* practice of automatically setting the scale. Note that when a network is to be analyzed a second time for the same frequency vector, the previous results, saved under the name *SAME*, can be used.




(a)



(b)

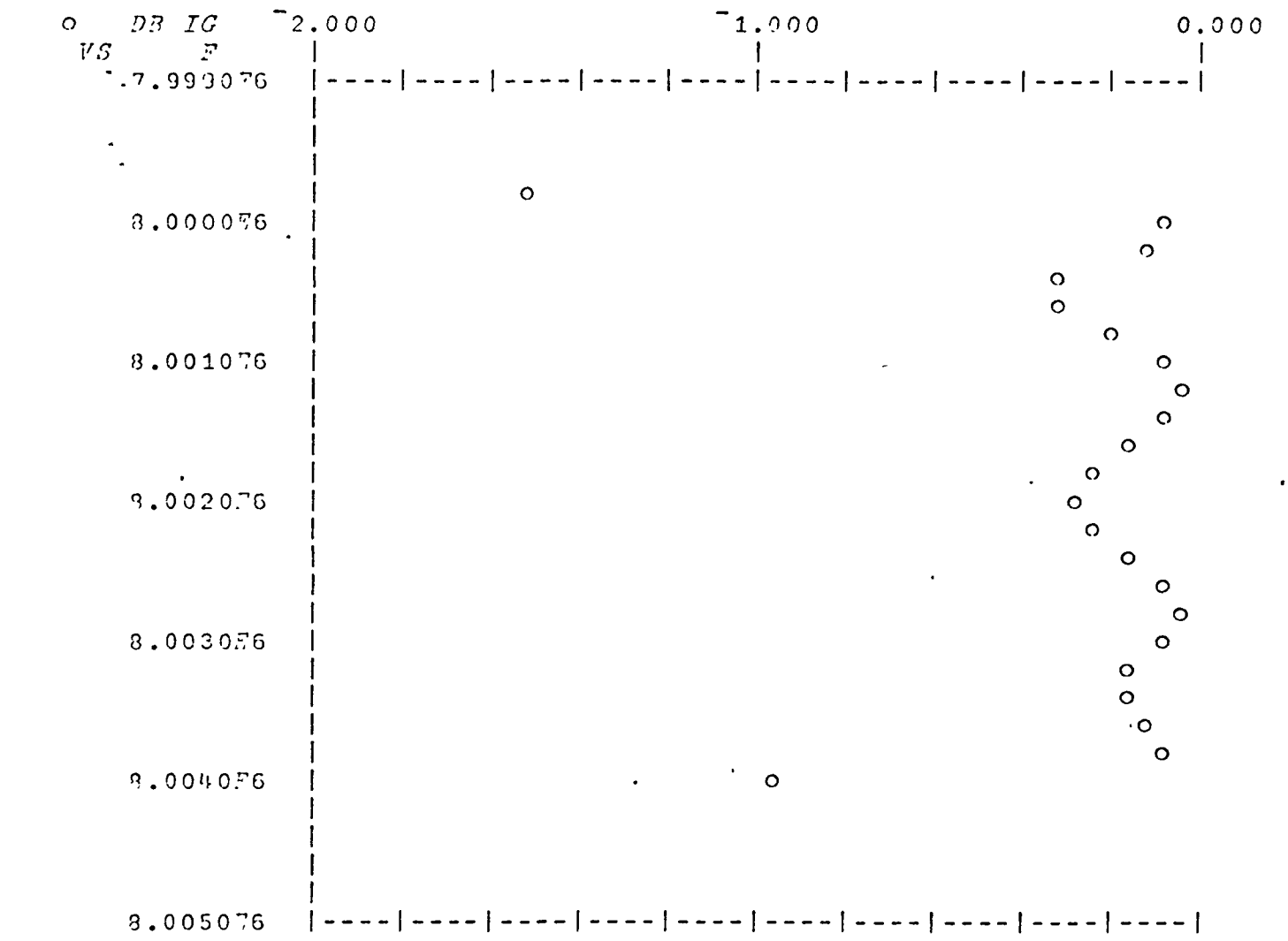
| | | |
|---|---|---|
| C1 | 14.94232 | pF. |
| C2 | 272.396 | pF. |
| C3 | 29.8748 | pF. |

(c)

Figure 10. (a) Crystal filter. (b) model for the crystals X1, X2, X3, and X4. (c) component values. The element values in the crystal models are not listed since they appear in *EXAMPLE* 2.

```
      A BIQUINTIC OF EXAMPLE 2.

      )LOAD 100 MARTHA
SAVED  14.32.54 04/23/73

      X1←(C 8.60776D¯12) P (C .01160137D¯12) S (L .0341232) S R .1752
      X2←(C 17.1140 D¯12) P (C .01773037D¯12) S (L .0223000) S R .1121
      X3←(C 12.7124 D¯12) P (C .01081484D¯12) S (L .0283568) S R .144
      X4←(C 20.0630 D¯12) P (C .01473529D¯12) S (L .0267554) S R .1345

      STAGE1←(C 14.34232D¯12) MC (WS X1) MPP (IT ¯1) MC WS X2
      STAGE2←(C 28.87480D¯12) MC (WS X3) MPP (IT ¯1) MC WS X4

      CRYSTALFILTER←STAGE1 MC (WS C 272.396D¯12) MC STAGE2

      ZG←500
      ZL←500
      F←7.999E6 + 0, 200×ι30


      PLOT 30 HIGH DB IG, DEG VG OF CRYSTALFILTER

CIRCUIT ANALYSIS BY MARTHA.   73.4   5/1/73  12:36
MARTHA COPYRIGHT (C) 1972 MASSACHUSETTS INSTITUTE OF TECHNOLOGY

o   DB IG    ¯2.000E1                ¯1.000E1                0.000
×  DEG VG    ¯2.000E2                 0.000                  2.000E2
 VS     F    |                        |                      |
    7.9990E6 |----o----|----|×---|----|----|----|----|----|----|
             |              ×
             |           ×     o
             |        ×              o
             |     ×                       o
             |                                   o
    8.0000E6 |                                    ×  o
             |                                 ×     o
             |                              ×        o
             |                           ×           o
             |                        ×              o
    8.0010E6 |                     ×                 o
             |                  ×                    o
             |               ×                       o
             |            ×                          o
             |         ×                             o
             |      ×                                o
    8.0020E6 |   ×                                   o
             |×                                      o
             |                                       o
             |                                       o
             |                                       o
    8.0030E6 |                                       o
             |                                       o
             |                                       o
             |                                       o
             |                                       o
             |                                       o
    8.0040E6 |                                    × o
             |                                 o×
             |                              o     ×
             |                           o        ×
             |                        o           ×
             |                     o              ×
             |                  o                 ×
    8.0050E6 |o----|----|----|----|----|----|---×|----|----|----|
```

```
      )COPY 100 MARTHA HSCALE
SAVED  14.33.35 04/23/73

      PLOT 30 HIGH -2 0 HSCALE DB IG OF SAT

CIRCUIT ANALYSIS BY MARTHA.   73*A   5/1/73  12:40
```



F=3.0027E6 + 200\130

PRINT DS IG, D?G VG, "AG ZI" OF CRYSTALLIZER

CIRCUIT ANALYSIS BY MARTHA.  73*A  5/1/73  12:42

| F | DB IG | D?G VG | "AG ZI" |
|---|---|---|---|
| 8.0022?6 | -2.5330?-1 | -1.7625E1 | 5.0901?2 |
| 8.002?"?3 | -1.7378?-1 | -3.2059?1 | 4.8032?2 |
| 8.0026?3 | -8.3978?-2 | -4.7233E1 | 4.9680?2 |
| 8.0023?3 | -4.7895E-2 | -6.3219?1 | 5.?129?2 |
| 8.0030?3 | -7.5248?-2 | -7.9333?1 | 5.0961F2 |
| | | | |
| 8.0032E6 | -1.4322E-1 | -9.7160F1 | 6.4241E2 |
| 8.0034?3 | -1.7420?-1 | -1.1538E2 | 6.0672?2 |
| 8.0036E6 | -1.0591E-1 | -1.3590E2 | 4.9115?2 |
| 8.0038E6 | -9.7414?-2 | -1.6144E2 | 3.3389?2 |
| 8.0040F6 | -9.5561E-1 | 1.6601E2 | 5.1276E2 |
| | | | |
| 8.0042F6 | -3.6975 | 1.3249E2 | 1.0933?3 |
| 8.0044F6 | -7.7434 | 1.0718E2 | 2.?858?3 |
| 8.0046?6 | -1.1990?1 | 9.0251?1 | 2.7834?4 |
| 8.0043?6 | -1.6035?1 | 7.8598E1 | 4.9476?3 |
| 8.0050?3 | -1.9844F1 | 7.0087E1 | 2.7733F3 |
| 8.0052?6 | -2.3469E1 | 6.3539?1 | 2.?697?3 |
| 8.0054?6 | -2.6981?1 | 5.8300E1 | 1.7124F3 |
| 8.0056E6 | -3.045"?1 | 5.3994?1 | 1.4930?3 |
| 8.0058?6 | -3.3939?1 | 5.0347?1 | 1.3495F3 |
| 8.0060?6 | -3.7632?1 | 4.7229?1 | 1.2432?3 |
| | | | |
| 8.0062?6 | -4.1602?1 | 4.4520F1 | 1.1621?3 |
| 8.0064?? | -4.6173E1 | 4.2143?1 | 1.0921F3 |
| 8.0066F6 | -5.2055F1 | 4.0048E1 | 1.0461?3 |
| 8.0069?6 | -6.2193?1 | 3.8267E1 | 1.0020?3 |
| 8.0070?6 | -6.7083?1 | -1.4392E2 | 9.?644?2 |
| | | | |
| 8.0072E6 | -5.7726E1 | -1.4527E2 | 9.?517F2 |
| 8.0074?6 | -5.4465?1 | -1.4666E2 | 9.0903?2 |
| 8.0076E6 | -5.2811?1 | -1.4794E2 | 8.?42?2 |
| 8.0078?6 | -5.1903E1 | -1.4913?2 | 8.?219?2 |
| 8.0080F6 | -5.1421E1 | -1.5023E2 | 8.4443F2 |

```
      F←8.00683E6 + 10×ι20

      PRINT DB IG, DEG VG OF CRYSTALFILTER

CIRCUIT ANALYSIS BY MARTHA.    73.4    5/1/73  12:46

     F           DB IG        DEG VG
---------- ----------- -----------
 8.0068E6   -6.3032E1     3.8194E1
 8.0068E6   -6.3944E1     3.8126E1
 8.0068E6   -6.4944E1     3.8063E1
 8.0068E6   -6.6054E1     3.8006E1
 8.0068E6   -6.7303E1     3.7960E1

 8.0069E6   -6.8736E1     3.7928E1
 8.0069E6   -7.0420E1     3.7919E1
 8.0069E6   -7.2471E1     3.7950E1
 8.0069E6   -7.5107E1     3.8063E1
 8.0069E6   -7.8828E1     3.8384E1

 8.0069E6   -8.5302E1    -3.9591E1
 8.0069E6   -1.0537E2    -1.6857E2
 8.0069E6   -8.4012E1    -1.4516E2
 8.0069E6   -7.8410E1    -1.4423E2
 8.0070E6   -7.5080E1    -1.4396E2

 8.0070E6   -7.2716E1    -1.4387E2
 8.0070E6   -7.0898E1    -1.4384E2
 8.0070E6   -6.9403E1    -1.4385E2
 8.0070E6   -6.8155E1    -1.4388E2
 8.0070E6   -6.7083E1    -1.4392E2


      )COPY 100 MARTHAT PLACES
SAVED  14.33.55 04/23/73

      F←8.00691E6 + ι10

      PRINT 7 PLACES DB IG, DEG VG OF CRYSTALFILTER

CIRCUIT ANALYSIS BY MARTHA.    73.4    5/1/73  12:49

     F             DB IG           DEG VG
------------ ------------- -------------
 8.0069117E6   -8.6210417E1    3.9833048E1
 8.0069127E6   -8.7446831E1    4.0263698E1
 8.0069137E6   -8.8756107E1    4.0757367E1
 8.0069147E6   -9.0231748E1    4.1466145E1
 8.0069157E6   -9.2152287E1    4.2509407E1

 8.0069167E6   -9.4512203E1    4.4196907E1
 8.0069177E6   -9.7735017E1    4.7413027E1
 8.0069187E6   -1.0277483E2    5.5770419E1
 8.0069197E6   -1.1131017E2    1.0387139E2
 8.0069207E6   -1.0536737E2    1.6256127E2

      ⍝ END OF EXAMPLE 2.
```

Next, it is desired to inspect the transition region and the stopband, so the frequency vector is redefined to cover the upper half of the passband and 4000 Hz above. The insertion gain, phase, and the magnitude of the input impedance, are printed.

An interesting phenomenon is observed at approximately 8.007 MHz. The insertion gain passes through a peak and the phase changes by 180 degrees. Apparently there is a transmission zero somewhere in the vicinity, and to view this in more detail, the frequency vector is redefined so as to expand the range between 8.0068 MHz and 8.0070 MHz. The depth of the notch in insertion gain (105 dB) and still rather sharp transition in phase suggest that the zero is very close to the $j\omega$ axis, and it is interesting to expand the passband again, between 8.00691 MHz and 8.00692 MHz. Note that the printing will not resolve these small changes in frequency so a larger number of significant figures is requested with the aid of the function *PLACES* from the *MARTHA* library. The frequency sweep is now in 1-Hz steps, and the high resolution available in this analysis is due to the fact that *MARTHA* (like APL) uses double-precision arithmetic.

C. Coaxial Low-Pass Filter

The third example, Figure 11, illustrates *MARTHA*'s ability to work with distributed as well as lumped networks. The original design of this seven-section low-pass filter was done by R. Levy and T. E. Rozzi[11], who stated that this filter is not particularly good except as an example of their design method, which they then used on a more practical, 23-section filter. Only the seven-section filter is analyzed here.

The coaxial discontinuity capacitances of the filter are important to its design. These capacitances are models which should be used at every junction between conductors of different size, as shown in Figure 12. *MARTHA* has a function named *COAXDISCAP* in the *MARTHA* library which calculates the values of these capacitors, when supplied with the appropriate radii of the line. Like all lengths in *MARTHA*, the radii must be given in meters, and so a conversion from the dimensions in inches in Figure 11 is necessary. In *EXAMPLE* 3, the first four discontinuity capacitors *C*1 through *C*4 are defined (the other four are equal because the filter is symmetric), and the conversion from inches to meters, and from diameters to radii, are done during the definitions (multiplication by 0.0254 meters per inch, and 0.5).

Next, the individual lengths of transmission line *L*1 through *L*4 are defined. In *MARTHA* a length of line is specified by its characteristic impedance and physical length. In our

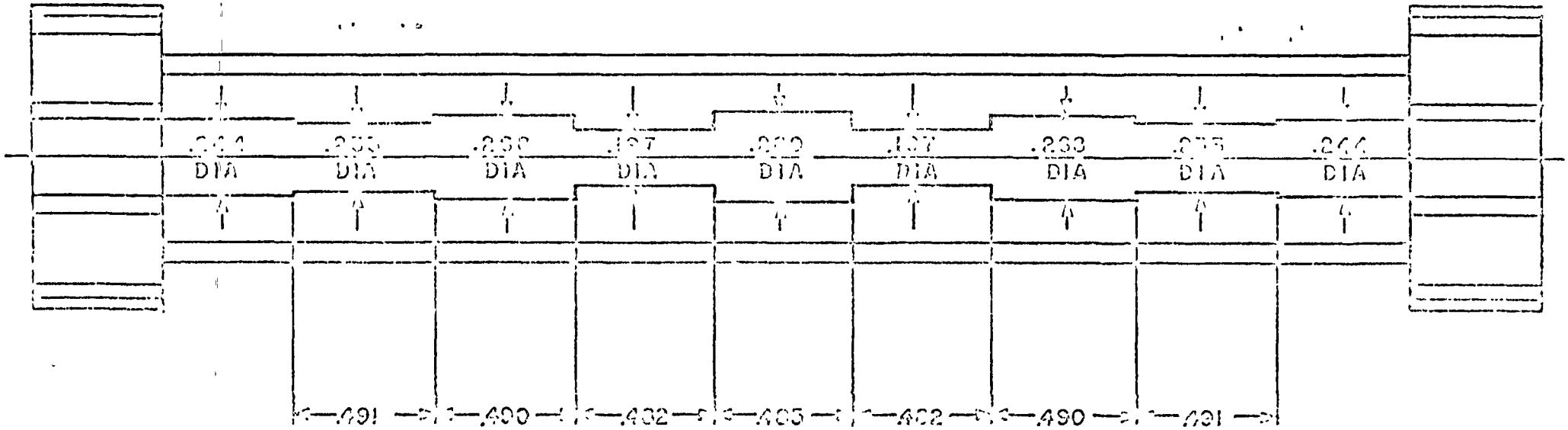





Figure 11. Seven-section coaxial low-pass filter. Dimensions are in inches. The diameter of the outer conductor is .561 inches, and the nominal characteristic impedance of the terminating lines is 50 ohms.

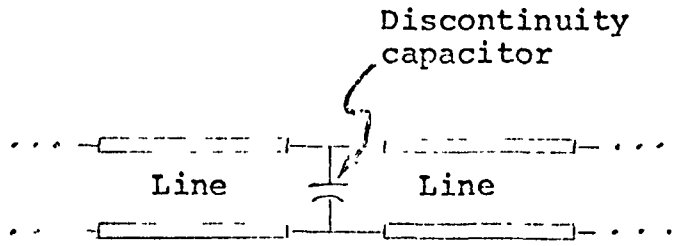
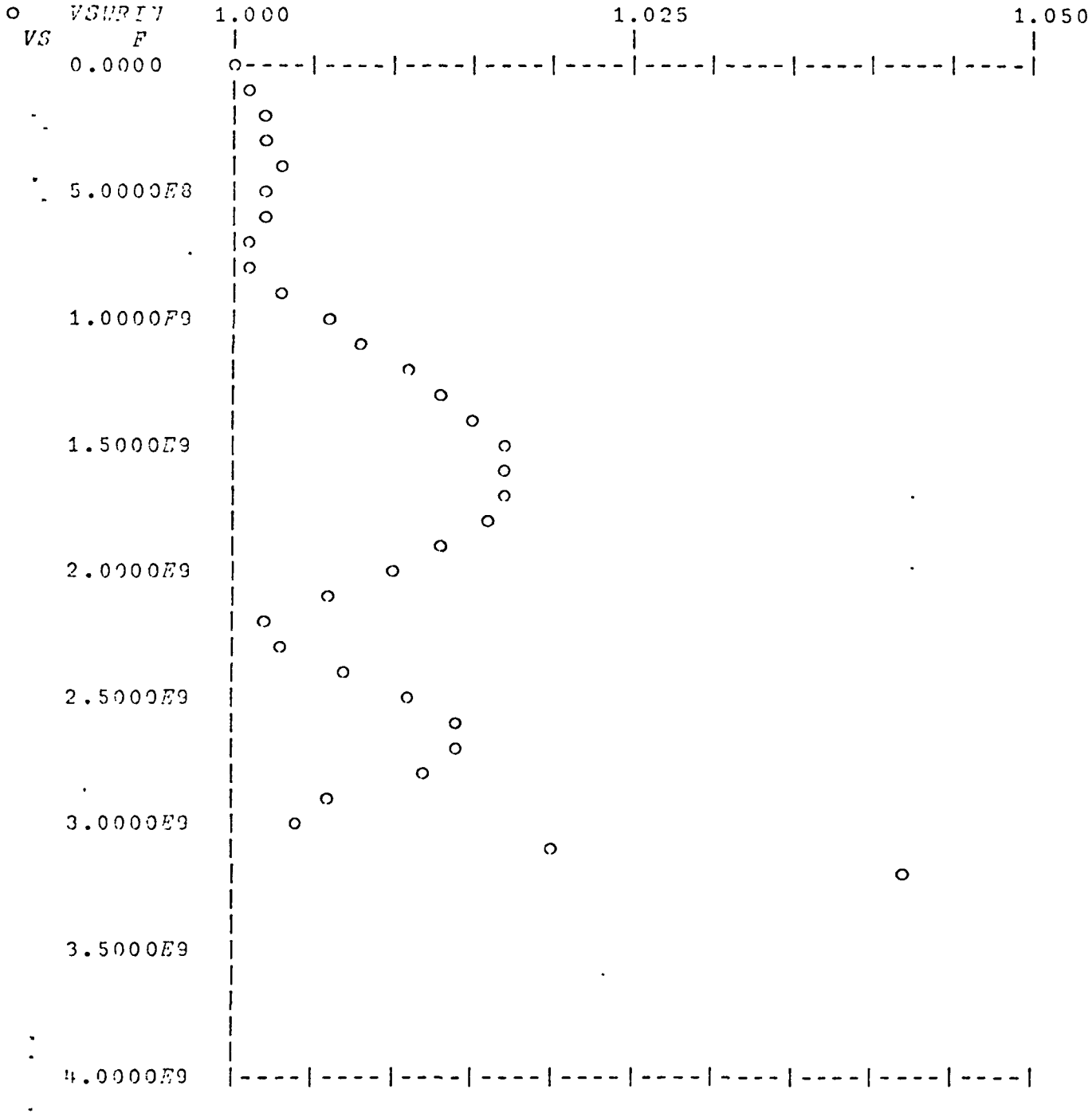



Figure 12. Discontinuity-capacitor model for the junction between coaxial lines of different size.

```
      A BEGINNING OF EXAMPLE 3.

      )LOAD 100 MARTHA
SAVED  14.32.54 04/23/73

      )COPY 100 MARTHAX COAXDISCAP
SAVED  14.36.46 04/23/73

      C1←C COAXDISCAP .5×.0254×.561 .244 .233
      C2←C COAXDISCAP .5×.0254×.561 .233 .268
      C3←C COAXDISCAP .5×.0254×.561 .268 .197
      C4←C COAXDISCAP .5×.0254×.561 .197 .289

      )COPY 100 MARTHAX COAX
SAVED  14.36.46 04/23/73

      L1←TEM COAX .0254×.2905 .1165 .491
      L2←TEM COAX .0254×.2905 .1340 .490
      L3←TEM COAX .0254×.2905 .0935 .482
      L4←TEM COAX .0254×.2805 .1445 .485

      LEFTSIDE←C1 WC L1 WC C2 WC L2 WC C3 WC L3 WC C4
      FILTER←LEFTSIDE WC L4 WC WH LEFTSIDE

      ZG←50
      ZL←50

      )COPY 100 MARTHAR VSWRIN
SAVED  14.35.57 04/23/73

      F←1E3×0.1 32
```

PLOT #0 HIGH VSWRIN OF FILTER

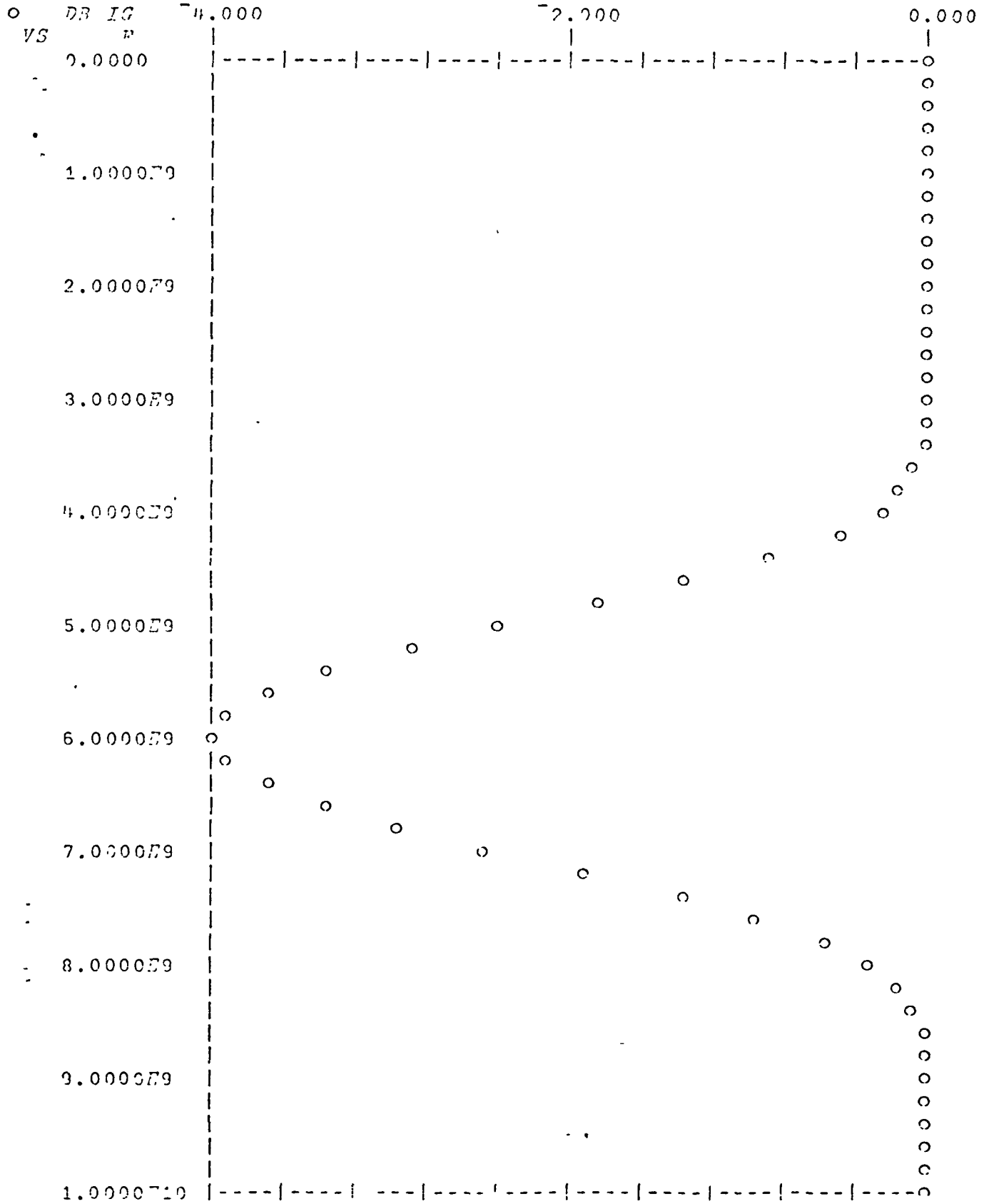
CIRCUIT ANALYSIS BY SPICE2A, 7301 5/1/73 12:57

F←0, 2^3×150

PLOT DB IG OF FILTER

CIRCUIT ANALYSIS BY MARSHA.  73.3  5/1/73  12:59




END OF EXAMPLE 3.

case we need to calculate the characteristic impedance from the dimensions. The function *COAX* in the *MARTHA* library is copied, and then used to do this. It expects as an argument a vector consisting of the inner and outer radii, and length, all in meters.

Since the filter is symmetric, it is possible to define the left side (consisting of the first three lines with the discontinuity capacitors placed at all junctions), and then use this definition twice in the final definition of the filter. Making the definition in this way eliminates the need to enter the same numerical data twice, and therefore tends to reduce keyboard errors.

The filter is designed to operate from a 50-ohm generator, and into a 50-ohm load, so the generator and load impedances *ZG* and *ZL* are set to 50. The VSWR at the input is wanted, and this response function *VSWRIN* is copied from the *MARTHA* library. The frequency vector is set, and a plot is requested. The result, of course, in in agreement with the analysis by Levy and Rozzi[11].

Filters of this sort are periodic, and have other passbands. In this case there is another passband centered around 12 GHz, so the stop band extends from 3 to 9 GHz. Levy and Rozzi stated that this seven-section filter had poor stopband attenuation, and the plot of the insertion gain up to 10 GHz reveals just how true this is. The maximum attenuation is only about 4 dB.

# VIII. LIMITATIONS

Every network-analysis program has two types of limitations not discussed so far. One has to do with computer resource limits, and the other with problems that are ill conditioned.

## A. Resource Limitations

Many programs have limitations on the number of nodes, or number of elements, or number of network defintiions allowed. These limitations arise from the amount of space set aside for certain variables. *MARTHA* has, for practical purposes, no such precise liritations since it is imbedded in APL, which uses dynamic storage for all variables. The resource limitations of *MARTHA* are somewhat more difficult to describe.

The primary limitation is caused by the finite size of APL workspaces (usually 32K bytes, each byte being 8 bits). The *MARTHA* functions require about 19K, leaving about 13K for the user's network definitions and terporary storage of results. If possible, *MARTHA* automatically analyzes with all frequencies at once; if this is not possible, it automatically selects a smaller number of frequencies, and repeats the analysis as many times as necessary, until all frequencies are accounted for. However, even this can be insufficient in some cases with lengthy output lists. Experience has shown

no trouble for analysis with up to fifty frequencies, provided not too much of the available space is taken by unrelated functions and data. The APL error message *WS FULL* indicates there is a problem of this sort. Some of the computers that carry *MAPTHA* have larger APL workspaces, in which case this ceases to be a real problem. If it is necessary to analyze with a large number of frequencies and available space cannot be created by erasing unnecessary objects, then *MARTHA* has a function entitled *ATATINE* which can be used to repeat the analysis for a small number of frequencies.

B. Ill-Conditioned Networks

Every network-analysis scheme has its own set of networks for which it performs poorly, or perhaps not at all. During analysis, *MARTHA* represents 1-port networks by their impedances and 2-port networks by their ABCD matrices. Thus *MAPTHA* is unable to handle networks for which these representations do not exist. For 1-port networks, this is only open circuits (or because of overflow or underflow problems, networks with impedance magnitudes greater than about $10^{37}$ or less than $10^{-37}$ ohms). For 2-port networks, this is any network for which the output voltage and current are related by one equation not containing the input voltage or current. This class of networks are those which produce no effect at the output when the input is excited, that is either those with disconnected

inputs and outputs, or those that are backwards unilateral. Generally when such a situation exists, it is easy to redefine the network so as to avoid the ill-conditioned case.

Certain element values, particularly zero, can lead to ill-conditioned networks, and Table 1 lists several such cases.

# IX. ERROR DIAGNOSTICS

*MARTHA* recognizes two classes of errors. One is errors so serious that *MARTHA* cannot reasonably proceed, and in the other case only some of the calculations may be in error and *MARTHA* continues.

For non-fatal errors, *MARTHA* prints a warning but continues. Examples include use of frequencies outside the range specified by the function *FLIMITS* and frequencies at which by coincidence the network is ill-conditioned. Generally in the latter case trouble is encountered when a denominator vanishes; the warning message is *ATTEMPT TO DIVIDE BY ZERO*. This also is printed when a user asks for a response function that is actually infinite, such as the admittance of a 0-ohm resistor. After this message, some of the results, but not all, may be in error.

Fatal errors are caught either by *MARTHA* or by APL. Among those caught by *MARTHA* are the wrong length for an argument for an element-definition function. Errors caught by *MARTHA* induce an explanatory message, usually indicating what was expected, followed by the message *MARTHA ERROR* and a diagnostic arrow pointing to the place where *MARTHA* got into trouble. These errors may often be regarded as a useful prompt, because by deliberately committing them, a user can remind himself what is expected. A complete list of these error messages appears in reference 2.

Errors not caught by *MARTHA*, but rather by APL, are reported by the message *NAME ERROR* where *NAME* is the name of one of the functions in *MARTHA*. This message may or may not be informative, but it is followed by an indication of the place where *MARTHA* got into trouble.

To recover from an error a user should type a single right arrow → followed by a carriage return, and then proceed after correcting the source of the error.

## X. REFERENCES

This chapter has not covered all aspects of using *MARTHA*; in particular, no attempt has been made to describe the *MARTHA* library fully, because it periodically gets additions. Complete instructions for using *MARTHA* appear in references 1 and 2 below. Each workspace in the *MARTHA* library has a variable entitled *DESCRIBE* which gives an up-to-date description of the contents. A succinct summary of *MARTHA* usage is available on-line in the workspace 100 *HOWMARTHA*.


1. P. Penfield Jr., "MARTHA User's Manual," The MIT Press, Cambridge, Massachusetts; 1971.

2. P. Penfield Jr., "MARTHA User's Manual, 1973 Addendum," The MIT Press, Cambridge, Massachusetts; 1973.

3. P. Penfield Jr., "Description of Electrical Networks Using Wiring Operators," Proc. IEEE, vol. 60, no. 1, pp. 49-53; January, 1972.

4. M. Greenspan, K. I. Thomassen, and P. Penfield Jr., "General-Purpose Microwave Circuit Analysis Incorporating Waveguide Discontinuity Models," Digest of Technical Papers, 1972 IEEE-GMTT International Microwave Symposium, Arlington Heights, Illinois; pp. 104-106; May 22-24, 1972.

5. B. B. Bhattacharyya, "Realization of an All-Pass Transfer Function," Proc. IEEE, vol. 57, no. 11, pp. 2092-2093; November, 1969.

6. J. L. Bordewijk, "Inter-reciprocity Applied to Electrical Networks," Applied Scientific Research, vol. B6, nos. 1-2, pp. 1-74; 1956.

7. S. W. Director and R. A. Rohrer, "The Generalized Adjoint Network and Network Sensitivities," IEEE Trans. on Circuit Theory, vol. CT-16, no. 3, pp. 318-323; August, 1969.

8. P. Penfield Jr., R. Spence, and S. Duinker, "Tellegen's Theorem and Electrical Networks," The MIT Press, Cambridge, Massachusetts; 1970; Appendix D.

9. N. Marcuvitz, "Waveguide Handbook," McGraw-Hill Book Co., Inc., New York, New York; 1951; Section 5.10.

10. S. Ramo, J. R. Whinnery, and T. Van Duzer, "Fields and Waves in Communication Electronics," John Wiley and Sons, Inc., New York, New York; 1965; Tables 5.14 and 8.09.

11. R. Levy and T. E. Rozzi, "Precise Design of Coaxial Low-Pass Filters," IEEE Trans. on Microwave Theory and Techniques, vol. MTT-16, no. 3, pp. 142-147; March, 1968.